YOUR £1.50

COMMODORE

SERIOUS USERS GUIDE 1988

THE ULTIMATE PROGRAM COLLECTION FOR C64 AND C128 OWNERS

INCLUDING:

DISK

PROGRAM

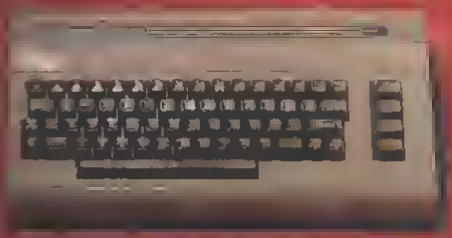
128 FO

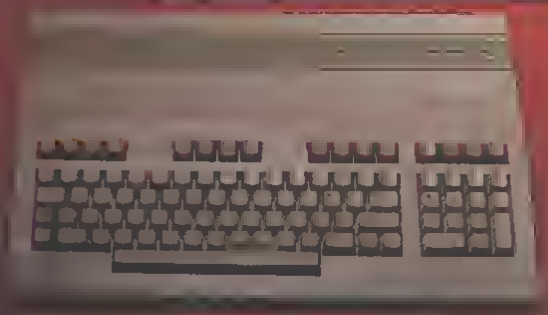
MESSAGE

GTX

DISK CATA

MOU





TECHNICAL INFORMATION FOR THE C64 AND C128

LIFESAVERS 1	C64	DISK/TAPE DEFAULT	1/1

This handy routine allows you to set the default device number to tape or disk for all loading and saving operations. Once it is set, you can then just type in LOAD "filename - you don't even need the final quotes.

If a machine code program is to be saved or loaded then the reset(SYS 720) and normal syntax will have to be used (eg ,8,1).

The program works by redirecting the load and save vectors at 816and 818 to this new routine at 679. To set the default to disk,use SYS 679 and to reset it to tape use SYS 708.

```
10 REM****************************
20 REM*      DEVICE SET          *
30 REM****************************
40 REM * SYS 679 :- DEFAULT TO
DISK *
50 REM * SYS 708 :- DEFAULT TO
TAPE *
60 REM * SYS 720 :- RESET TO
NORMAL *
70 REM****************************
80 FOR L=679 TO 758:READ A:POKE
L,A:C=C+A:NEXT
90 IF C<>8969 THEN PRINT "DATA
ERROR":END
100 DATA 169,8,141,230,2,141,241,
2,169,229,141,48,3,169,240,141
110 DATA 50,3,169,2,141,49,3,169,
2,141,51,3,96,32,175,2
120 DATA 169,1,141,230,2,141,
241,2,96,169,165,141,48,3,169,244
130 DATA 141,49,3,169,237,141,50,
3,169,245,141,51,3,96,169,8
140 DATA 133,186,169,0,133,10,76,
165,244,169,8,133,186,76,237,245
```

LIFESAVERS 2	C64	LISTING PAUSE	1/1

The problem with the Commodore 64 operating system is that a listed program shoots up the screen too quickly to read. This leaves the user having to laboriously list lines in groups of about ten.

This program solves the problem by allowing the user to pause the scrolling program by pressing the spacebar. The routine is placed high in the 49152 block of memory at 53200 and therefore leaves plenty of scope for using programs which call up other code routines in this area.

Type in the program and save it. When you think you may wish to use it, load it in before doing anything else with the computer and RUN. It will automatically execute itself so all you then have to do is to remember to press the spacebar.

```
10 C=0:REM LIST STOP BY STEPHEN
ELMER
20 FOR I=53200 TO 53245:READ
A:C=C+A:POKE I,A:NEXT
30 IF C<>6879 THEN PRINT"ERROR IN
DATA":END
40 SYS 53200
50 DATA 169,219,141,38,3,169,207,
141
60 DATA 39,3,96,72,165,204,201,1
70 DATA 208,24,165,197,201,60,
208,18
80 DATA 201,60,208,250,165,197,
201,60
90 DATA 240,250,104,76,202,241
```

YOUR
COMMODORE SERIOUS USERS GUIDE 1988

Editor:
Stuart Cooke
Deputy Editor
Eric Doyle
Typesetting
Magazine Typesetters
Design
Wakeworth Design
Artist
Alan Batchelor

Within these pages you will find information to help any serious C64 or C128 user to get the best from their computer. Utilities for programmers and other computer users are backed-up by an informative technical section and an extensive hints and tips guide.

If your forte is Basic programming the GTX Compiler can convert a program to run at over 30 times its normal speed.

A program also needs style if it is going to impress anyone and the 128 Font Editor and the Message Construction Kit for the C64 provide easy routes to adding a 'designer look' to your routines.

Instead of attaching a printer or disk drive to the serial port, using it to link through to another C64, C128, Plus 4 or C16 can add a new dimension to games playing. With the Bus Route 64 program you possess a key which opens up the world of interactive, two player games.

On the technical side there are memory maps of the C128, C64 and 1541 disk drive with detailed tables of many more vital statistics in a quick reference format.

The Your Commodore Serious Users Guide is more than a magazine, it's a reference guide that deserves a place beside every Commodore 64 or 128.

Contents

Your Commodore is a monthly magazine appearing on the first Friday of each month. Argus Specialist Publications Limited. Editorial & Advertisement Office: Your Commodore, No 1 Golden Square, London W1R 3AB. Telephone 01-437 0626. Telex 8811896. Subscription rates upon application to Your Commodore Subscriptions Department, Infonet Ltd, 5 River Park Estate, Berkhamsted, Herts HP4 1HL. U.S.A. Subscription Agent: Wise Owl Worldwide Publications, 4314 West 238th Street, Torrance CA 90505 U.S.A. Distribution: SM Distribution, 6 Leigham Court Road, London SW16 2PG. Printed by Chase Web, Plymouth.

Listings

Get it right first time with our deluxe program system for the C64

You may have noticed that our listings are free of those horrible little black blobs which send you searching around the keyboard for a suitable graphic symbol. You may also have noticed the funny numbers by the side of each line of the listing. Fret no more, it's all part of our easy entry aid.

Instead of those nasty graphics and rows of countless spaces in PRINT statements and strings we use a special coding system. The code, or mnemonic, is always contained in square brackets and you'll soon learn to decipher their meanings.

For example, [SA] would mean type in a Shifted A, or in the case of spades in layman's terms, and [SA10] would mean a row of ten of these symbols.

[S+2] means hold down the shift key and press the plus key twice. It doesn't take a great leap of logic to realise that [C+2] means exactly the same thing except that the Commodore key (bottom left of the keyboard) is held down instead of the shift key.

If more than two spaces appear in a statement then this will be printed as [SPC4] or, exceptionally, [SSPC4]. Translated into English this means press the spacebar four times or in the latter case hold the shift key down while you do it.

A string of special characters could appear as

[CTRL N, DOWN2,LEFT5,BLUE, F3,C3]

This would be achieved by holding down the CTRL key as you press N, press the cursor key down twice, the cursor left key five times, press the key marked BLUE while holding down the CTRL key, press the F3 key and, finally hold the Commodore key down while pressing the number two key (C2 would of course make the computer print in brown).

Always remember that you should only have a row of graphics characters on your screen with no square brackets and no commas, unless something like this appears

[SS],[C*]

In this case the two characters should have a comma between them.

On rare occasions [REV T] will appear in a listing. This is a delete symbol and is created by entering the line up to this mnemonic. Then type a closing quotation mark (SHIFT & 2) and delete it. This gets the computer out of quotes mode. Hold down CTRL and press the number nine key (RVSON), type the relevant number of reversed T's and then hold down CTRL and press zero (RVSOFF). Next type another quotation mark and delete it again. Now finish the line and press RETURN.

A list of these special cases is given in the table but remember that only one of these mnemonics will appear outside of a PRINT string, the symbol for pi. This may appear when its value is needed in a calculation so this may look something like

CC=2*[PI]*R

Ignore the square brackets and just type in a shifted upward pointing arrow (ie. the pi symbol).

PROGRAM: SYNTAX CHECKER

```
5 REM SYTAX CHECKER - ERIC DOYLE

10 BL=10    LN=70    SA=49152
20 FOR L=0 TO BL CX=0 FOR O=0 TO
 15
30 READ A IF A>255THENPRINT"NUMB
ER TO LARGE" LN+(L*10) STOP
40 CX=CX+A POKE SA+L*16+O A NEXT
 O
50 READ A IF A<>CX THENPRINT"ERR
OR IN LINE" LN+(L*10) STOP
60 NEXT L SYS 49152 NEW
70 DATA 173 5 3,201 166 208 31 1
20 169 8 141 32 208 141 33 208 1
6N7
80 DATA 169 7 141 134 2 169 13 3
2 210 255 169 54 141 4 3 169 160
2
90 DATA 152 141 5 3 88 96 120 16
9 164 141 4 3 169 165 141 5 3566

100 DATA 3 169 14 141 134 2 141
32 208 169 6 141 33 208 88 96 15
85
110 DATA 32 124 165,76 138 72 15
2 76 162 0 165 20 133 254 165 21
 1747
120 DATA 24 101 254,133 254 169
0,6 240 18,89 254 133 254 232 18
8 8246
130 DATA 0,2 240 8 24 101 254 13
3 254 232 208 233 169 1 141 134
6134
140 DATA 6 165 254,74 74 74 74,3
6 156 138 32 210,255 165,254 41
6054
150 DATA 15 32 156 188 32 210,25
5 169 13 32 210 255 169 13 38,21
0 1995
160 DATA 255,169 7 141 134 6 104
 168 104,170 104 96 24 105 48 20
1 1832
170 DATA 58,16 1 96 24 105 7 96
0 0 0,0 0 0 0 403
```

by Eric Doyle

Mnemonic	Symbol	Keypress
[RIGHT]		CRSR left/right
[LEFT]		SHIFT & CRSR left/right
[DOWN]		CRSR up/down
[UP]		SHIFT & CRSR up/down
[F1]		f1 key
[F2[		SHIFT & f1 key
[F3]		f3 key
[F4]		SHIFT & f3 key
[F5]		f5 key
[F6]		SHIFT & f5 key
[F7]		f7 key
[F8]		SHIFT & f7 key
[HOME]		CLR/HOME
[CLR]		SHIFT & CLR/HOME
[RVSON]		CTRL & 9
[RVSOFF]		CTRL & 0

Mnemonic	Symbol	Keypress
[BLACK]		CTRL & 1
[WHITE]		CTRL & 2
[RED]		CTRL & 3
[CYAN]		CTRL & 4
[PURPLE]		CTRL & 5
[GREEN]		CTRL & 6
[BLUE]		CTRL & 7
[YELLOW]		CTRL & 8
[POUND]		£
[LARROW]		←
[UPARROW]		↑
[PI]		SHIFT & ↑
[INST]		SHIFT & INST/DEL
[REV T]		see text
[Cletter]		CBM + letter
[Sletter]		SHIFT + letter

Checksum Program

The hexadecimal numbers appearing in a column to the left of the listing should not be typed in with the program. These are merely checksum values and are there to help you get each line right. Don't worry if you don't understand the hexadecimal system, as long as you can compare two characters on the screen with the corresponding two characters in the magazine you can use our line checking program.

Type in the Checksum Program, make sure that you've not made any mistakes and save it to tape or disk immediately because it will be used with most of the present and future listings appearing in Your Commodore.

At the start of each programming session, load Checksum and run it. The screen will turn brown with yellow characters and each time you type in a line and press the RETURN key a number will appear on the screen in white. This should be the same as the corresponding value in the magazine.

If the two values don't relate to one another, you have not copied the line exactly as printed so go back and check each character carefully. When you find the error simply correct it and press RETURN again.

If you want to turn off the checker simply type SYS49152 and the screen will return to the familiar blue colours. You can then do whatever it was you wanted to do and if this doesn't use the area where Checksum lies you can go back to it with the same SYS command.

No system is foolproof but the chances of two errors cancelling one another out are so remote that we believe our listings are more reliable than any other magazine in the world. So get typing!

Mouse

Type-in this listing for C64 mice and see how they run!

by W I Sellers

The Neos mouse included in the Commodore Connoisseur's Collection is extremely good at its job. It has two operating modes and can act as a joystick emulator to make it widely compatible with commercial software, or in its standard mode, which allows much smoother control. The main drawback is the weak documentation explaining how the mouse can be incorporated in your own programs. The following routine enables you to read the position of the mouse from Basic.

How It Works

The mouse transmits its movement as an X and Y displacement. The program is a machine-code routine which receives the movement and assigns the values automatically to two Basic variables, DX and DY. The status of the left-hand button is returned in the variable FB. These variable names are created within the routine, so they don't need to be previously assigned.

To use the mouse, switch off the normal function of the keyboard and call the mouse set-up routine at decimal 50003. The mouse-read routine at decimal 50000 must be called each time its position needs to be updated. DX and DY contain the distance moved in the X and Y direction since the last time the read routine was called.

For machine-code users, the subroutine that performs the mouse read is at $C3A3 to $C42C with the X and Y displacements and button status stored from $C42D to $C42F. The rest of the program is concerned with assigning the values to the Basic variables.

In Use

The program is in the form of a simple demonstration with the mouse used to move a sprite round the screen. Lines 470–920 form a subroutine that creates the machine-code routine and it's this bit that needs to be incorporated into other programs.

You need to insert the mouse after the program has been started because the keyboard produces nonsense with the mouse in place. Save a copy of the program before running it in case there are any mistakes that might lead to a crash. The RUN/STOP key is disabled by the program so that the program can only be stopped by RUN/STOP and RESTORE being pressed simultaneously.

```
LINE# LOC  OBJECT LABELS      LINE

   10 0000                    ;PAGE ZERO LABELS
   20 0002                    ;
   30 0002          NMVE      =$FB
   40 0002          ADDRESS   =$FC
   50 0002          BION      =$55
   60 0002                    ;
   70 0002                    ;BASIC ROM ROUTINES
   80 0002                    ;
   90 0002          FINDVAR   =$B08D
  100 0002          CONVERTYPELP =$B8AD
  110 0002          FLPTOMEM  =$B8F7
  120 0002                    ;
  130 0002                    ;
  140 0002                    ;  INPUT AND ADDRESSES
  150 0002                    ;
  160 0002          PADDLEX   =$D419
  170 0002          PORTB     =$DC00
  180 0002          DDRB      =$DC02
  190 0002          ICR1      =$DC0D
  200 0002          CIA2      =$DD00
  210 0002          CRAI      =$DC0E
  220 0002          CRBI      =$DC0F
  230 0002          CRAE      =$DD0E
  240 0002          CRBE      =$DD0F
  250 0002          ;
  260 0002          ;START

  270 0002
  280 C350                    *
  290 C350                    ;  SOOOO
  300 C350                    ;
  310 C350                    ;JUMP TABLE
  320 C350                    ;
  330 C350          4C3BC3 READ   JMP START
  340 C353          4C3DC4 SETUP  JMP INITIALIZE
  350 C356                    ;
  360 C356                    ;MAIN ROUTINE TO READ MOUSE
  370 C356                    ;AND PUT RESULTS IN 3 VARIABLES
  380 C356                    ;  BX  BY AND FB
  390 C356                    ;
  400 C356          20C3C3 START  JSR MOUSEREAD
  410 C359                    ;
  420 C359                    ;  DEAL WITH DELX
  430 C359                    ;
  440 C359          A6FC      LDX DELX
  450 C35B          A4FD      LDY DELX+1
  460 C35D          8A        TXA
  470 C35E          20D2BD    JSR $BDD2
  480 C361                    ;
  490 C361                    ;DEAL WITH DELY
  500 C361                    ;
  510 C363          A6FE      LDX DELY
  520 C365          A4FF      LDY DELY+1
  530 C367          8A        TXA
```

```
LINE# LOC  OBJECT LABELS     LINE
  940 C38A 2078C9            JSR STOREACC
  950 C38D                   
  960 C38D                   ;REAL WITH BUTTON
  970 C38D                   
  980 C38D A9FC41            LDA BUTTON
  990 C370 4F45              LDX # F
 1000 C372 A041              LDY # 0
 1010 C374 2078C3            JSR STOREACC
 1020 C377                   
 1030 C377 EO                RTN
 1040 C378                   
 1050 C378                   
 1060 C378                   ;ROUTINE TO STORE ACCUMULATOR
 1070 C378                    ON 8001C VARIABLE IN KY
 1080 C378                   
 1090 C378                    STORE REGISTERS
 1100 C378                   
 1110 C378 48       STOREACC PHA
 1120 C379 8546              STX NAME
 1130 C37B 8447              STY NAME+1
 1140 C37D                   
 1150 C37D                   ;PUSH DUMMY VALUE ON STACK
 1160 C37D A9FF              LDA #0FF
 1170 C37F 48                PHA
 1180 C380                   
 1190 C380                   ;FIND VARIABLE LOCATION
 1200 C380                   
 1210 C380 20B7D0            JSR FINDVAR
 1220 C383                   
 1230 C383                   ;GET RID OF DUMMY
 1240 C383                   
 1250 C383 68                PLA
 1260 C384                   
 1270 C384                   ;CONVERT A (2 S COMPLEMENT) TO
 1280 C384                   ;FLOATING POINT
 1290 C384                   
 1300 C384 68                PLA
 1310 C385                   "
 1320 C385                   ;CHECK IF POSITIVE
 1330 C385                   
 1340 C385 10DA              BPL DA1
 1350 C387                   
 1360 C387                    CONVERT NEGATIVE NUMBER
 1370 C387                   
 1380 C387 49FF              EOR #11111111
 1390 C389 AA                TAY
 1400 C38A E8                INY
 1410 C38B 20AE88            JSR CONVERTYINFLP
 1420 C38E 4CE8C3            JMP DA2
 1430 C391                   
 1440 C391                   ;CONVERT POSITIVE NUMBER
 1450 C391                   
 1460 C391 AA       DA1      TAY
 1470 C392 20AE83            JSR CONVERTYINFLP
 1480 C395 A586              LDA S1BB
 1490 C397 49FF              EOR #11111111
 1500 C399 8586              STA S1BA
 1510 C39B                   
 1520 C39B                    STORE FLOATING POINT ACCUMULATOR
 1530 C39B                   
 1540 C3EB A97F     DA2      LDX #ACCSTOR
 1550 C39D A000              LDY #ACCSTOR*
 1560 C39F 20D4BB            JSR FLPTOMEM
 1570 C3A2                   
 1580 C3A2                    RETURN
 1590 C3A2                   
 1600 C3A2 EE                RTN
 1610 C3A3                   
 1620 C3A3                    ROUTINE TO READ PAUSE POSITION
 1630 C3A3                   
 1640 C3A3                    DISABLE INTERRUPTS
 1650 C3A3                   
 1660 C3A3 78       PAUSEDOWN SEI
 1670 C3A4                   
 1680 C3A4                    STORE IN/OUT REGISTERS
 1690 C3A4                   
 1700 C3A4 A600CC            LDA PORT2
 1710 C3A7 48                PHA
 1720 C3A8 A000              LDA #0000
 1730 C3AB 48                PHA
 1740 C3AC                   
 1750 C3AC                   ;SET BIT 4 TO OUTPUT
 1760 C3AC                   
 1770 C3AC A816              LDA #816
 1780 C3AE 8D00BC            STA 0BBBB
 1790 C3B1                   
 1800 C3B1                   ;CLEAR BIT 4
 1810 C3B1                   
 1820 C3B1 A900CC            LDA PORT2
 1830 C3B4 49EF              AND #0EF
 1840 C3B6 8D00CC            STA PORT2
 1850 C3B9                   
 1860 C3B9                    PAUSE 60 CYCLES
 1870 C3B9                   
 1880 C3B9 A03C              LDY #03C
 1890 C3BB 20A0C3            JSR PAUSE
 1900 C3BE                   
 1910 C3BE                   ;GET LOW NYBBLE
 1920 C3BE                   
 1930 C3BE A900CC            LDA PORT2
 1940 C3C1 09F0              ORA #0F0
 1950 C3C3 8D00CC            STA PORT2
 1960 C3C6                   
 1970 C3C6                    WAIT FOR BUTTON PRESS
 1980 C3C6                   
 1990 C3C6 A9FC      WAIT1   LDA #FC1
 2000 C3C8 8DFFCC            STA 0CCFF
 2010 C3CB                   
 2020 C3CB                    GET NEXT INSTRUCTION
 2030 C3CB                   
 2040 C3CB                    BYTE   0BC
 2050 C3CC                   
 2060 C3CC A900      FC1     LDA #000
 2070 C3CE 8D00CC            STA 0CC00
 2080 C3D1                   
 2090 C3D1                    RECOVER IN/OUT REGISTERS
 2100 C3D1                   
 2110 C3D1 A900CC            LDA PORT2
 2120 C3D4 2900              AND #000
 2130 C3D6 8D00CC            STA PORT2
 2140 C3D9 68                PLA
 2150 C3DA 8500              STA 0000
 2160 C3DC 68                PLA
 2170 C3DD 8501              STA 0001
 2180 C3DF                   
 2190 C3DF                    ENABLE INTERRUPTS AND RETURN
 2200 C3DF                   
 2210 C3DF 58                CLI
 2220 C3E0 60                RTS
 2230 C3E1                   
 2240 C3E1                   ;ROUTINE TO PAUSE Y * 16 CYCLES
 2250 C3E1                   
 2260 C3E1 EA       PAUSE    NOP
 2270 C3E2 EA                NOP
 2280 C3E3 EA                NOP
 2290 C3E4 88                DEY
 2300 C3E5 D0FA              BNE PAUSE
 2310 C3E7 60                RTS
 2320 C3E8                   
 2330 C3E8                   ;INITIALIZE MOUSE READER
```

```
LINE# LOC  OBJECT LABELS     LINE
 1000 C3E8 EA                NOP A
 1010 C3E9 EA                NOP A
 1020 C3EA EA                NOP A
 1030 C3EB 8D00CC            STA 0CLX
 1040 C3EE                   
                              SET BIT 4
 1050 C3EE                   
 1060 C3EE A900CC            LDA PORT2
 1070 C3F1 0900              ORA #010
 1080 C3F3 8D00CC            STA PORT2
 1090 C3F6                   
 1100 C3F6                    PAUSE 50 CYCLES
 1110 C3F6                   
 1120 C3F6 A03C              LDY #03C
 1130 C3F8 20A0C3            JSR PAUSE
 1140 C3FB                   
                              GET LOW NYBBLE
 1150 C3FB                   
 1160 C3FB A900CC            LDA PORT2
 1170 C3FE 0900              ORA #010
 1180 C400 8D00CC            STA PORT2
 1190 C403                   
                              SET BIT 4
 1200 C403                   
 1210 C403 A900CC            LDA PORT2
 1220 C406 0900              ORA #010
 1230 C408 8D00CC            STA PORT2
 1240 C40B                   
                              PAUSE 50 CYCLES
 1250 C40B                   
 1260 C40B A03C              LDY #03C
 1270 C40D 20A0C3            JSR PAUSE
 1280 C410                   
                              GET LOW NYBBLE
 1290 C410                   
 1300 C410 A900CC            LDA PORT2
 1310 C413 2900              AND #00F
 1320 C415 8D00CC            STA PORT2
 1330 C418                   
                              READ BUTTON
 1340 C418                   
 1350 C418 AD01CC            LDA PADDLEX
 1360 C41B C9FF              CMP #0FF
 1370 C41D D0A1              BNE FC1
 1380 C41F                   
                              WAIT IN A IF BUTTON PRESSED
 1390 C41F                   
 1400 C41F 18                CLC
 1410 C420 60                RTS
 1420 C421                   
                              BYTE   0BC
 1430 C421                   
 1440 C421 A900      FC1     LDA #000
 1450 C423 8DFFCC            STA 0CCFF
 1460 C426                   
                              RECOVER IN/OUT REGISTERS
 1470 C426                   
 1480 C426 68                PLA
 1490 C427 8500              STA 0000
 1500 C429 68                PLA
 1510 C42A 8501              STA 0001
 1520 C42D                   
                              ENABLE INTERRUPTS AND RETURN
 1530 C42D                   
 1540 C42D 58                CLI
 1550 C42E 60                RTS
 1560 C42F                   
 1570 C42F                   ;INITIALIZE MOUSE READER
 1580 C42F                   
 1590 C42F A900      INITMOUSE LDA #000
 1600 C431 8DFFCC            STA 0CCFF
 1610 C434                   
                              SET BIT 4
 1620 C434                   
 1630 C434 A900CC            LDA PORT2
 1640 C437 0900              ORA #010
 1650 C439 8D00CC            STA PORT2
 1660 C43C                   
                              PAUSE 50 CYCLES
 1670 C43C                   
 1680 C43C A03C              LDY #03C
 1690 C43E 20A0C3            JSR PAUSE
 1700 C441                   
                              GET LOW NYBBLE
 1710 C441                   
 1720 C441 A900CC            LDA PORT2
 1730 C444 2900              AND #00F
 1740 C446 8D00CC            STA PORT2
 1750 C449                   
                              SET BIT 4
 1760 C449                   
 1770 C449 A900CC            LDA PORT2
 1780 C44C 0900              ORA #010
 1790 C44E 8D00CC            STA PORT2
 1800 C451                   
                              PAUSE 50 CYCLES
 1810 C451                   
 1820 C451 A03C              LDY #03C
 1830 C453 20A0C3            JSR PAUSE
 1840 C456                   
                              GET LOW NYBBLE
 1850 C456                   
 1860 C456 A900CC            LDA PORT2
 1870 C459 2900              AND #00F
 1880 C45B 8D00CC            STA PORT2
 1890 C45E                   
                              SET BIT 4
 1900 C45E                   
 1910 C45E A900CC            LDA PORT2
 1920 C461 0900              ORA #010
 1930 C463 8D00CC            STA PORT2
 1940 C466                   
                              PAUSE 50 CYCLES
 1950 C466                   
 1960 C466 A03C              LDY #03C
 1970 C468 20A0C3            JSR PAUSE
 1980 C46B                   
                              GET LOW NYBBLE
 1990 C46B                   
 2000 C46B A900CC            LDA PORT2
 2010 C46E 2900              AND #00F
 2020 C470 8D00CC            STA PORT2
 2030 C473                   
                              READ BUTTON
 2040 C473                   
 2050 C473 AD01CC            LDA PADDLEX
 2060 C476 C9FF              CMP #0FF
 2070 C478 D0A7              BNE FC1
 2080 C47A                   
                              WAIT IN A IF BUTTON PRESSED
 2090 C47A                   
 2100 C47A 18                CLC
 2110 C47B 60                RTS
 2120 C47C                   
                              BYTE   0BC
 2130 C47C                   
 2140 C47C A900      FC1     LDA #000
 2150 C47E 8DFFCC            STA 0CCFF
 2160 C481                   
                              RECOVER IN/OUT REGISTERS
 2170 C481                   
 2180 C481 A900CC            LDA PORT2
 2190 C484 2900              AND #000
 2200 C486 8D00CC            STA PORT2
 2210 C489 68                PLA
 2220 C48A 8500              STA 0000
 2230 C48C 68                PLA
 2240 C48D 8501              STA 0001
 2250 C490                   
                              ENABLE INTERRUPTS AND RETURN
 2260 C490                   
 2270 C490 58                CLI
 2280 C491 60                RTS
 2290 C492                   
 2300 C492                   ;INITIALIZE MOUSE READER
 2310 C492                   
 2320 C492 A900      INITMOUSE LDA #000
 2330 C494 8DFFCC            STA 0CCFF
 2340 C497                   
                              SET BIT 4
 2350 C497                   
 2360 C497 A900CC            LDA PORT2
 2370 C49A 0900              ORA #010
 2380 C49C 8D00CC            STA PORT2
 2390 C49F                   
                              PAUSE 50 CYCLES
 2400 C49F                   
 2410 C49F A03C              LDY #03C
 2420 C4A1 20A0C3            JSR PAUSE
 2430 C4A4                   
                              GET LOW NYBBLE
 2440 C4A4                   
 2450 C4A4 A900CC            LDA PORT2
 2460 C4A7 2900              AND #00F
 2470 C4A9 8D00CC            STA PORT2
 2480 C4AC                   
                              SET BIT 4
 2490 C4AC                   
 2500 C4AC A900CC            LDA PORT2
 2510 C4AF 0900              ORA #010
 2520 C4B1 8D00CC            STA PORT2
 2530 C4B4                   
                              PAUSE 50 CYCLES
 2540 C4B4                   
 2550 C4B4 A03C              LDY #03C
 2560 C4B6 20A0C3            JSR PAUSE
 2570 C4B9                   
                              GET LOW NYBBLE
 2580 C4B9                   
 2590 C4B9 A900CC            LDA PORT2
 2600 C4BC 2900              AND #00F
 2610 C4BE 8D00CC            STA PORT2
 2620 C4C1                   
                              READ BUTTON
 2630 C4C1                   
 2640 C4C1 AD01CC            LDA PADDLEX
 2650 C4C4 C9FF              CMP #0FF
 2660 C4C6 D0B4              BNE FC1
 2670 C4C8                   
                              WAIT IN A IF BUTTON PRESSED
 2680 C4C8                   
 2690 C4C8 18                CLC
 2700 C4C9 60                RTS
 2710 C4CA                   
                              BYTE   0BC
 2720 C4CA                   
 2730 C4CA A900      FC1     LDA #000
 2740 C4CC 8DFFCC            STA 0CCFF
 2750 C4CF                   
                              RECOVER IN/OUT REGISTERS
 2760 C4CF                   
 2770 C4CF A900CC            LDA PORT2
 2780 C4D2 2900              AND #000
 2790 C4D4 8D00CC            STA PORT2
 2800 C4D7 68                PLA
 2810 C4D8 8500              STA 0000
 2820 C4DA 68                PLA
 2830 C4DB 8501              STA 0001
 2840 C4DD                   
                              ENABLE INTERRUPTS AND RETURN
 2850 C4DD                   
 2860 C4DD 58                CLI
 2870 C4DE 60                RTS
 2880 C4DF                   
 2890 C4DF                   ;INITIALIZE MOUSE READER
 2900 C4DF                   
 2910 C4DF A900      INITMOUSE LDA #000
 2920 C4E1 8DFFCC            STA 0CCFF
 2930 C4E4                   
                              SET BIT 4
 2940 C4E4                   
 2950 C4E4 A900CC            LDA PORT2
 2960 C4E7 0900              ORA #010
 2970 C4E9 8D00CC            STA PORT2
 2980 C4EC                   
                              PAUSE 50 CYCLES
 2990 C4EC                   
 3000 C4EC A03C              LDY #03C
 3010 C4EE 20A0C3            JSR PAUSE
 3020 C4F1                   
                              GET LOW NYBBLE
 3030 C4F1                   
 3040 C4F1 A900CC            LDA PORT2
 3050 C4F4 2900              AND #00F
 3060 C4F6 8D00CC            STA PORT2
 3070 C4F9                   
                              READ BUTTON
 3080 C4F9                   
 3090 C4F9 AD01CC            LDA PADDLEX
 3100 C4FC C9FF              CMP #0FF
 3110 C4FE D0A7              BNE FC1
 3120 C500                   
                              WAIT IN A IF BUTTON PRESSED
 3130 C500                   
 3140 C500 18                CLC
 3150 C501 60                RTS
 3160 C502                   
                              BYTE   0BC
 3170 C502                   
 3180 C502 A900      FC1     LDA #000
 3190 C504 8DFFCC            STA 0CCFF
 3200 C507                   
                              RECOVER IN/OUT REGISTERS
 3210 C507                   
 3220 C507 A900CC            LDA PORT2
 3230 C50A 2900              AND #000
 3240 C50C 8D00CC            STA PORT2
 3250 C50F 68                PLA
 3260 C510 8500              STA 0000
 3270 C512 68                PLA
 3280 C513 8501              STA 0001
 3290 C515                   
                              ENABLE INTERRUPTS AND RETURN
 3300 C515                   
 3310 C515 58                CLI
 3320 C516 60                RTS
 3330 C517                   
 3340 C517                   ;INITIALIZE MOUSE READER
 3350 C517                   
 3360 C517 A900      INITMOUSE LDA #000
 3370 C519 8DFFCC            STA 0CCFF
 3380 C51C                   
                              SET BIT 4
 3390 C51C                   
 3400 C51C A900CC            LDA PORT2
 3410 C51F 0900              ORA #010
 3420 C521 8D00CC            STA PORT2
 3430 C524                   
                              PAUSE 50 CYCLES
 3440 C524                   
 3450 C524 A03C              LDY #03C
 3460 C526 20A0C3            JSR PAUSE
 3470 C529                   
                              GET LOW NYBBLE
 3480 C529                   
 3490 C529 A900CC            LDA PORT2
 3500 C52C 2900              AND #00F
 3510 C52E 8D00CC            STA PORT2
 3520 C531                   
                              READ BUTTON
 3530 C531                   
 3540 C531 AD01CC            LDA PADDLEX
 3550 C534 C9FF              CMP #0FF
 3560 C536 D0A7              BNE FC1
 3570 C538                   
                              WAIT IN A IF BUTTON PRESSED
 3580 C538                   
 3590 C538 18                CLC
 3600 C539 60                RTS
 3610 C53A                   
                              BYTE   0BC
 3620 C53A                   
 3630 C53A A900      FC1     LDA #000
 3640 C53C 8DFFCC            STA 0CCFF
 3650 C53F                   
                              RECOVER IN/OUT REGISTERS
 3660 C53F                   
 3670 C53F A900CC            LDA PORT2
 3680 C542 2900              AND #000
 3690 C544 8D00CC            STA PORT2
 3700 C547 68                PLA
 3710 C548 8500              STA 0000
 3720 C54A 68                PLA
 3730 C54B 8501              STA 0001
 3740 C54D                   
                              ENABLE INTERRUPTS AND RETURN
 3750 C54D                   
 3760 C54D 58                CLI
 3770 C54E 60                RTS
 3780 C54F                   
 3790 C54F                   ;INITIALIZE MOUSE READER
 3800 C54F                   
 3810 C54F A900      INITMOUSE LDA #000
 3820 C551 8DFFCC            STA 0CCFF
 3830 C554                   
                              SET BIT 4
 3840 C554                   
 3850 C554 A900CC            LDA PORT2
 3860 C557 0900              ORA #010
 3870 C559 8D00CC            STA PORT2
 3880 C55C                   
                              PAUSE 50 CYCLES
 3890 C55C                   
 3900 C55C A03C              LDY #03C
 3910 C55E 20A0C3            JSR PAUSE
 3920 C561                   
                              GET LOW NYBBLE
 3930 C561                   
 3940 C561 A900CC            LDA PORT2
 3950 C564 2900              AND #00F
 3960 C566 8D00CC            STA PORT2
 3970 C569                   
                              READ BUTTON
 3980 C569                   
 3990 C569 AD01CC            LDA PADDLEX
 4000 C56C C9FF              CMP #0FF
 4010 C56E D0A7              BNE FC1
 4020 C570                   
                              WAIT IN A IF BUTTON PRESSED
 4030 C570                   
 4040 C570 18                CLC
 4050 C571 60                RTS
 4060 C572                   
                              BYTE   0BC
 4070 C572                   
 4080 C572 A900      FC1     LDA #000
 4090 C574 8DFFCC            STA 0CCFF
 4100 C577                   
                              RECOVER IN/OUT REGISTERS
 4110 C577                   
 4120 C577 A900CC            LDA PORT2
 4130 C57A 2900              AND #000
 4140 C57C 8D00CC            STA PORT2
 4150 C57F 68                PLA
 4160 C580 8500              STA 0000
 4170 C582 68                PLA
 4180 C583 8501              STA 0001
 4190 C585                   
                              ENABLE INTERRUPTS AND RETURN
 4200 C585                   
 4210 C585 58                CLI
 4220 C586 60                RTS
 4230 C587                   
 4240 C587                   ;INITIALIZE MOUSE READER
 4250 C587                   
 4260 C587 A900      INITMOUSE LDA #000
 4270 C589 8DFFCC            STA 0CCFF
 4280 C58C                   
                              SET BIT 4
 4290 C58C                   
 4300 C58C A900CC            LDA PORT2
 4310 C58F 0900              ORA #010
 4320 C591 8D00CC            STA PORT2
 4330 C594                   
                              PAUSE 50 CYCLES
 4340 C594                   
 4350 C594 A03C              LDY #03C
 4360 C596 20A0C3            JSR PAUSE
 4370 C599                   
                              GET LOW NYBBLE
 4380 C599                   
 4390 C599 A900CC            LDA PORT2
 4400 C59C 2900              AND #00F
 4410 C59E 8D00CC            STA PORT2
 4420 C5A1                   
                              READ BUTTON
 4430 C5A1                   
 4440 C5A1 AD01CC            LDA PADDLEX
 4450 C5A4 C9FF              CMP #0FF
 4460 C5A6 D0A7              BNE FC1
 4470 C5A8                   
                              WAIT IN A IF BUTTON PRESSED
 4480 C5A8                   
 4490 C5A8 18                CLC
 4500 C5A9 60                RTS
 4510 C5AA                   
                              BYTE   0BC
 4520 C5AA                   
 4530 C5AA A900      FC1     LDA #000
 4540 C5AC 8DFFCC            STA 0CCFF
 4550 C5AF                   
                              RECOVER IN/OUT REGISTERS
 4560 C5AF                   
 4570 C5AF A900CC            LDA PORT2
 4580 C5B2 2900              AND #000
 4590 C5B4 8D00CC            STA PORT2
 4600 C5B7 68                PLA
 4610 C5B8 8500              STA 0000
 4620 C5BA 68                PLA
 4630 C5BB 8501              STA 0001
 4640 C5BD                   
                              ENABLE INTERRUPTS AND RETURN
 4650 C5BD                   
 4660 C5BD 58                CLI
 4670 C5BE 60                RTS
 4680 C5BF                   
 4690 C5BF                   ;INITIALIZE MOUSE READER
 4700 C5BF                   
 4710 C5BF A900      INITMOUSE LDA #000
 4720 C5C1 8DFFCC            STA 0CCFF
 4730 C5C4                   
                              SET BIT 4
 4740 C5C4                   
 4750 C5C4 A900CC            LDA PORT2
 4760 C5C7 0900              ORA #010
 4770 C5C9 8D00CC            STA PORT2
 4780 C5CC                   
                              PAUSE 50 CYCLES
 4790 C5CC                   
 4800 C5CC A03C              LDY #03C
 4810 C5CE 20A0C3            JSR PAUSE
 4820 C5D1                   
                              GET LOW NYBBLE
 4830 C5D1                   
 4840 C5D1 A900CC            LDA PORT2
 4850 C5D4 2900              AND #00F
 4860 C5D6 8D00CC            STA PORT2
 4870 C5D9                   
                              READ BUTTON
 4880 C5D9                   
 4890 C5D9 AD01CC            LDA PADDLEX
 4900 C5DC C9FF              CMP #0FF
 4910 C5DE D0A7              BNE FC1
 4920 C5E0                   
                              WAIT IN A IF BUTTON PRESSED
 4930 C5E0                   
 4940 C5E0 18                CLC
 4950 C5E1 60                RTS
 4960 C5E2                   
                              BYTE   0BC
 4970 C5E2                   
 4980 C5E2 A900      FC1     LDA #000
 4990 C5E4 8DFFCC            STA 0CCFF
 5000 C5E7                   
                              RECOVER IN/OUT REGISTERS
 5010 C5E7                   
 5020 C5E7 A900CC            LDA PORT2
 5030 C5EA 2900              AND #000
 5040 C5EC 8D00CC            STA PORT2
 5050 C5EF 68                PLA
 5060 C5F0 8500              STA 0000
 5070 C5F2 68                PLA
 5080 C5F3 8501              STA 0001
 5090 C5F5                   
                              ENABLE INTERRUPTS AND RETURN
 5100 C5F5                   
 5110 C5F5 58                CLI
 5120 C5F6 60                RTS
 5130 C5F7                   
 5140 C5F7                   ;INITIALIZE MOUSE READER
 5150 C5F7                   
 5160 C5F7 A900      INITMOUSE LDA #000
 5170 C5F9 8DFFCC            STA 0CCFF
 5180 C5FC                   
                              SET BIT 4
 5190 C5FC                   
 5200 C5FC A900CC            LDA PORT2
 5210 C5FF 0900              ORA #010
 5220 C601 8D00CC            STA PORT2
 5230 C604                   
                              PAUSE 50 CYCLES
 5240 C604                   
 5250 C604 A03C              LDY #03C
 5260 C606 20A0C3            JSR PAUSE
 5270 C609                   
                              GET LOW NYBBLE
 5280 C609                   
 5290 C609 A900CC            LDA PORT2
 5300 C60C 2900              AND #00F
 5310 C60E 8D00CC            STA PORT2
 5320 C611                   
                              READ BUTTON
 5330 C611                   
 5340 C611 AD01CC            LDA PADDLEX
 5350 C614 C9FF              CMP #0FF
 5360 C616 D0A7              BNE FC1
 5370 C618                   
                              WAIT IN A IF BUTTON PRESSED
 5380 C618                   
 5390 C618 18                CLC
 5400 C619 60                RTS
 5410 C61A                   
                              BYTE   0BC
 5420 C61A                   
 5430 C61A A900      FC1     LDA #000
 5440 C61C 8DFFCC            STA 0CCFF
 5450 C61F                   
                              RECOVER IN/OUT REGISTERS
 5460 C61F                   
 5470 C61F A900CC            LDA PORT2
 5480 C622 2900              AND #000
 5490 C624 8D00CC            STA PORT2
 5500 C627 68                PLA
 5510 C628 8500              STA 0000
 5520 C62A 68                PLA
 5530 C62B 8501              STA 0001
 5540 C62D                   
                              ENABLE INTERRUPTS AND RETURN
 5550 C62D                   
 5560 C62D 58                CLI
 5570 C62E 60                RTS
 5580 C62F                   
 5590 C62F                   ;INITIALIZE MOUSE READER
 5600 C62F                   
 5610 C62F A900      INITMOUSE LDA #000
 5620 C631 8DFFCC            STA 0CCFF
 5630 C634                   
                              SET BIT 4
 5640 C634                   
 5650 C634 A900CC            LDA PORT2
 5660 C637 0900              ORA #010
 5670 C639 8D00CC            STA PORT2
 5680 C63C                   
                              PAUSE 50 CYCLES
 5690 C63C                   
 5700 C63C A03C              LDY #03C
 5710 C63E 20A0C3            JSR PAUSE
 5720 C641                   
                              GET LOW NYBBLE
 5730 C641                   
 5740 C641 A900CC            LDA PORT2
 5750 C644 2900              AND #00F
 5760 C646 8D00CC            STA PORT2
 5770 C649                   
                              READ BUTTON
 5780 C649                   
 5790 C649 AD01CC            LDA PADDLEX
 5800 C64C C9FF              CMP #0FF
 5810 C64E D0A7              BNE FC1
 5820 C650                   
                              WAIT IN A IF BUTTON PRESSED
 5830 C650                   
 5840 C650 18                CLC
 5850 C651 60                RTS
 5860 C652                   
                              BYTE   0BC
 5870 C652                   
 5880 C652 A900      FC1     LDA #000
 5890 C654 8DFFCC            STA 0CCFF
 5900 C657                   
                              RECOVER IN/OUT REGISTERS
 5910 C657                   
 5920 C657 A900CC            LDA PORT2
 5930 C65A 2900              AND #000
 5940 C65C 8D00CC            STA PORT2
 5950 C65F 68                PLA
 5960 C660 8500              STA 0000
 5970 C662 68                PLA
 5980 C663 8501              STA 0001
 5990 C665                   
                              ENABLE INTERRUPTS AND RETURN
 6000 C665                   
 6010 C665 58                CLI
 6020 C666 60                RTS
 6030 C667                   
 6040 C667                   ;INITIALIZE MOUSE READER
 6050 C667                   
 6060 C667 A900      INITMOUSE LDA #000
 6070 C669 8DFFCC            STA 0CCFF
 6080 C66C                   
                              SET BIT 4
 6090 C66C                   
 6100 C66C A900CC            LDA PORT2
 6110 C66F 0900              ORA #010
 6120 C671 8D00CC            STA PORT2
 6130 C674                   
                              PAUSE 50 CYCLES
 6140 C674                   
 6150 C674 A03C              LDY #03C
 6160 C676 20A0C3            JSR PAUSE
 6170 C679                   
                              GET LOW NYBBLE
 6180 C679                   
 6190 C679 A900CC            LDA PORT2
 6200 C67C 2900              AND #00F
 6210 C67E 8D00CC            STA PORT2
 6220 C681                   
                              READ BUTTON
 6230 C681                   
 6240 C681 AD01CC            LDA PADDLEX
 6250 C684 C9FF              CMP #0FF
 6260 C686 D0A7              BNE FC1
 6270 C688                   
                              WAIT IN A IF BUTTON PRESSED
 6280 C688                   
 6290 C688 18                CLC
 6300 C689 60                RTS
 6310 C68A                   
                              BYTE   0BC
 6320 C68A                   
 6330 C68A A900      FC1     LDA #000
 6340 C68C 8DFFCC            STA 0CCFF
 6350 C68F                   
                              RECOVER IN/OUT REGISTERS
 6360 C68F                   
 6370 C68F A900CC            LDA PORT2
 6380 C692 2900              AND #000
 6390 C694 8D00CC            STA PORT2
 6400 C697 68                PLA
 6410 C698 8500              STA 0000
 6420 C69A 68                PLA
 6430 C69B 8501              STA 0001
 6440 C69D                   
                              ENABLE INTERRUPTS AND RETURN
 6450 C69D                   
 6460 C69D 58                CLI
 6470 C69E 60                RTS
 6480 C69F                   
 6490 C69F                   ;INITIALIZE MOUSE READER
 6500 C69F                   
 6510 C69F A900      INITMOUSE LDA #000
 6520 C6A1 8DFFCC            STA 0CCFF
 6530 C6A4                   
                              SET BIT 4
 6540 C6A4                   
 6550 C6A4 A900CC            LDA PORT2
 6560 C6A7 0900              ORA #010
 6570 C6A9 8D00CC            STA PORT2
 6580 C6AC                   
                              PAUSE 50 CYCLES
 6590 C6AC                   
 6600 C6AC A03C              LDY #03C
 6610 C6AE 20A0C3            JSR PAUSE
 6620 C6B1                   
                              GET LOW NYBBLE
 6630 C6B1                   
 6640 C6B1 A900CC            LDA PORT2
 6650 C6B4 2900              AND #00F
 6660 C6B6 8D00CC            STA PORT2
 6670 C6B9                   
                              READ BUTTON
 6680 C6B9                   
 6690 C6B9 AD01CC            LDA PADDLEX
 6700 C6BC C9FF              CMP #0FF
 6710 C6BE D0A7              BNE FC1
 6720 C6C0                   
                              WAIT IN A IF BUTTON PRESSED
 6730 C6C0                   
 6740 C6C0 18                CLC
 6750 C6C1 60                RTS
 6760 C6C2                   
                              BYTE   0BC
 6770 C6C2                   
 6780 C6C2 A900      FC1     LDA #000
 6790 C6C4 8DFFCC            STA 0CCFF
 6800 C6C7                   
                              RECOVER IN/OUT REGISTERS
 6810 C6C7                   
 6820 C6C7 A900CC            LDA PORT2
 6830 C6CA 2900              AND #000
 6840 C6CC 8D00CC            STA PORT2
 6850 C6CF 68                PLA
 6860 C6D0 8500              STA 0000
 6870 C6D2 68                PLA
 6880 C6D3 8501              STA 0001
 6890 C6D5                   
                              ENABLE INTERRUPTS AND RETURN
 6900 C6D5                   
 6910 C6D5 58                CLI
 6920 C6D6 60                RTS
 6930 C6D7                   
 6940 C6D7                   ;INITIALIZE MOUSE READER
 6950 C6D7                   
 6960 C6D7 A900      INITMOUSE LDA #000
 6970 C6D9 8DFFCC            STA 0CCFF
 6980 C6DC                   
                              SET BIT 4
 6990 C6DC                   
 7000 C6DC A900CC            LDA PORT2
 7010 C6DF 0900              ORA #010
 7020 C6E1 8D00CC            STA PORT2
 7030 C6E4                   
                              PAUSE 50 CYCLES
 7040 C6E4                   
 7050 C6E4 A03C              LDY #03C
 7060 C6E6 20A0C3            JSR PAUSE
 7070 C6E9                   
                              GET LOW NYBBLE
 7080 C6E9                   
 7090 C6E9 A900CC            LDA PORT2
 7100 C6EC 2900              AND #00F
 7110 C6EE 8D00CC            STA PORT2
 7120 C6F1                   
                              READ BUTTON
 7130 C6F1                   
 7140 C6F1 AD01CC            LDA PADDLEX
 7150 C6F4 C9FF              CMP #0FF
 7160 C6F6 D0A7              BNE FC1
 7170 C6F8                   
                              WAIT IN A IF BUTTON PRESSED
 7180 C6F8                   
 7190 C6F8 18                CLC
 7200 C6F9 60                RTS
 7210 C6FA                   
                              BYTE   0BC
 7220 C6FA                   
 7230 C6FA A900      FC1     LDA #000
 7240 C6FC 8DFFCC            STA 0CCFF
 7250 C6FF                   
                              RECOVER IN/OUT REGISTERS
 7260 C6FF                   
 7270 C6FF A900CC            LDA PORT2
 7280 C702 2900              AND #000
 7290 C704 8D00CC            STA PORT2
 7300 C707 68                PLA
 7310 C708 8500              STA 0000
 7320 C70A 68                PLA
 7330 C70B 8501              STA 0001
 7340 C70D                   
                              ENABLE INTERRUPTS AND RETURN
 7350 C70D                   
 7360 C70D 58                CLI
 7370 C70E 60                RTS
 7380 C70F                   
 7390 C70F                   ;INITIALIZE MOUSE READER
 7400 C70F                   
 7410 C70F A900      INITMOUSE LDA #000
 7420 C711 8DFFCC            STA 0CCFF
 7430 C714                   
                              SET BIT 4
 7440 C714                   
 7450 C714 A900CC            LDA PORT2
 7460 C717 0900              ORA #010
 7470 C719 8D00CC            STA PORT2
 7480 C71C                   
                              PAUSE 50 CYCLES
 7490 C71C                   
 7500 C71C A03C              LDY #03C
 7510 C71E 20A0C3            JSR PAUSE
 7520 C721                   
                              GET LOW NYBBLE
 7530 C721                   
 7540 C721 A900CC            LDA PORT2
 7550 C724 2900              AND #00F
 7560 C726 8D00CC            STA PORT2
 7570 C729                   
                              READ BUTTON
 7580 C729                   
 7590 C729 AD01CC            LDA PADDLEX
 7600 C72C C9FF              CMP #0FF
 7610 C72E D0A7              BNE FC1
 7620 C730                   
                              WAIT IN A IF BUTTON PRESSED
 7630 C730                   
 7640 C730 18                CLC
 7650 C731 60                RTS
 7660 C732                   
                              BYTE   0BC
 7670 C732                   
 7680 C732 A900      FC1     LDA #000
 7690 C734 8DFFCC            STA 0CCFF
 7700 C737                   
                              RECOVER IN/OUT REGISTERS
 7710 C737                   
 7720 C737 A900CC            LDA PORT2
 7730 C73A 2900              AND #000
 7740 C73C 8D00CC            STA PORT2
 7750 C73F 68                PLA
 7760 C740 8500              STA 0000
 7770 C742 68                PLA
 7780 C743 8501              STA 0001
 7790 C745                   
                              ENABLE INTERRUPTS AND RETURN
 7800 C745                   
 7810 C745 58                CLI
 7820 C746 60                RTS
 7830 C747                   
 7840 C747                   ;INITIALIZE MOUSE READER
 7850 C747                   
 7860 C747 A900      INITMOUSE LDA #000
 7870 C749 8DFFCC            STA 0CCFF
 7880 C74C                   
                              SET BIT 4
 7890 C74C                   
 7900 C74C A900CC            LDA PORT2
 7910 C74F 0900              ORA #010
 7920 C751 8D00CC            STA PORT2
 7930 C754                   
                              PAUSE 50 CYCLES
 7940 C754                   
 7950 C754 A03C              LDY #03C
 7960 C756 20A0C3            JSR PAUSE
 7970 C759                   
                              GET LOW NYBBLE
 7980 C759                   
 7990 C759 A900CC            LDA PORT2
 8000 C75C 2900              AND #00F
 8010 C75E 8D00CC            STA PORT2
 8020 C761                   
                              READ BUTTON
 8030 C761                   
 8040 C761 AD01CC            LDA PADDLEX
 8050 C764 C9FF              CMP #0FF
 8060 C766 D0A7              BNE FC1
 8070 C768                   
                              WAIT IN A IF BUTTON PRESSED
 8080 C768                   
 8090 C768 18                CLC
 8100 C769 60                RTS
 8110 C76A                   
                              BYTE   0BC
 8120 C76A                   
 8130 C76A A900      FC1     LDA #000
 8140 C76C 8DFFCC            STA 0CCFF
 8150 C76F                   
                              RECOVER IN/OUT REGISTERS
 8160 C76F                   
 8170 C76F A900CC            LDA PORT2
 8180 C772 2900              AND #000
 8190 C774 8D00CC            STA PORT2
 8200 C777 68                PLA
 8210 C778 8500              STA 0000
 8220 C77A 68                PLA
 8230 C77B 8501              STA 0001
 8240 C77D                   
                              ENABLE INTERRUPTS AND RETURN
 8250 C77D                   
 8260 C77D 58                CLI
 8270 C77E 60                RTS
 8280 C77F                   
 8290 C77F                   ;INITIALIZE MOUSE READER
 8300 C77F                   
 8310 C77F A900      INITMOUSE LDA #000
 8320 C781 8DFFCC            STA 0CCFF
 8330 C784                   
                              SET BIT 4
 8340 C784                   
 8350 C784 A900CC            LDA PORT2
 8360 C787 0900              ORA #010
 8370 C789 8D00CC            STA PORT2
 8380 C78C                   
                              PAUSE 50 CYCLES
 8390 C78C                   
 8400 C78C A03C              LDY #03C
 8410 C78E 20A0C3            JSR PAUSE
```

```
LINE# LOC  OBJECT LABELS          LINE

0986 C430
0970 C430                    ;DISABLE INTERRUPTS
0980 C430
0990 C430
0000 C430 78      INITIALIZE  SEI
0010 C431                     ;
0020 C431                     ; CLEAR INTERRUPT MASKS ON CIA 1
0030 C431                     ;
0040 C431 A97F                LDA #$7F
0050 C433 8D0D0C              STA $DC01
0060 C436 8D0D00              STA $DC0D
0070 C436                     ;
0080 C436                     ;SET PORT 10 OUTPUT
0090 C436                     ;
0100 C438 A9FF                LDA #$FF
0110 C43A 8D02DC              STA $DD02
0120 C43D A97F                LDA #$7F

LINE# LOC  OBJECT LABELS          LINE

0130 C43F 8D00DC              STA $DC00
0140 C442                     ;
0150 C442                     ; SET TIMERS TO DEFAULT VALUES
0160 C442                     ;
0170 C442 A900                LDA #$00
0180 C444 8D04DC              STA $DC04
0190 C447 8D05DC              STA $DC05
0200 C44A 8D0E00              STA $DC0E
0210 C44D 8D0F00              STA $DC0F
0220 C450                     ;
0230 C450                     ;ENABLE INTERRUPTS AND RETURN
0240 C450                     ;
0250 C450 58                  CLI
0260 C451 60                  RTS
0270 C452
0280 C452                     END
```

```
PROGRAM: MOUSE MANAGER

10 REM MOUSE CODE -THIS IS E
SSENTIAL IF THE SAVE ROUTINE
 IS USED!!!!
20 REM LINES 10-70 CAN BE OM
ITTED IF YOU DON'T WANT TO S
AVE THE CODE
30 FORA=0TO82:READX:NEXT:GOS
UB 550
40 POKE193,80:POKE194,165:PO
KE197,83:POKE175,156
50 POKE 187,7:POKE186,8
60 POKE 183,10:POKE 186,0:RE
M POKE 185,1 FOR CASSETTE
70 POKE 185,0:SYS62594
80 REM MOUSE MANAGER DEMO FR
OGRAM
90 REM (C)OPYRIGHT 1987 W I
SELLERS
100 REM CLEAR SCREEN
110 PRINT"{CLR}POKING IN DAT
A{3 ."
120 REM POKE IN SPRITE DATA
130 RESTORE:FORS=832TO894:RE
ADA:POKES,A:NEXT
140 REM POKE IN MACHINE CODE
150 GOSUB550
160 PRINT"{CLR}INSERT MOUSE
IN JOYSTICK PORT2"
170 PRINT"{DOWN}AND PRESS <S
PACE> WHEN READY"
180 GETA$:IFA$<>" "THENI80
190 PRINT"{CLR}", POKE53281,
0
200 REM INITIALIZE MOUSE REA
DER
210 SYS50003
220 REM INITIALIZE SPRITE D
230 POKE2040,13
240 V=53246
250 POKEV+21,1
260 REM INITIALIZE POSITION
270 X=160:Y=100 C=0
280 REM READ MOUSE
290 SYS50000:PRINTOX,OY
300 REM OX=X CHANGE OY=Y CHA
NGE
310 REM F6=1 IF PRESSED
320 X=X+OX Y=Y+OY
330 REM CHECK X AND Y LIMITS
340 IFX<OTHENX=0
350 IFX>511THENX=511
360 IFY<OTHENY=0
370 IFY>255THENY=255
380 IFC>255THENC=0
390 REM CALCULATE X VALUES
400 X2=INT(X/256) X1=X-256*X
2
410 C=C+F6 IFC=256THENC=0
420 POKEV,X1 POKEV+1,Y POKEV
+16,X2
430 IF(CAND15)=OTHENC=C+1
440 POKEV+39,C
450 GOTO250
460 REM SPRITE DATA
470 DATA0,0,0,0,0,0,0,0
480 DATA0,0,0,0,0,0 0,8
490 DATA0 8,28,0,28,60,0,30
500 DATA126 34 63,127,192,25
5,127,255
510 DATA255,124,73,31,120,12
7,15,112
520 DATA62,7,48,20,5,15,0,4
530 DATA8,0,8,0,0,0,0,0
540 DATA0 0,0,0,0,0,0
550 REM FROM MOUSE READER
ROUTINE
560 REM FROM #C350 TO #C452
570 D=50000 L=600
580 T=0
590 FORX=1TO8
600 READA IFA=-1THEN660
610 POKED,A D=D+1 T=T+A
620 POKE53280,A NEXT
630 READA IFA=-1THEN660
640 IFI<>ATHENPRINT"ERROR IN
 LINE "L END
650 L=L+10 GOTO580
660 IFO<>50258+ITHENPRINT"AD
DRESS ERROR" END
670 RETURN
680 DATA76,00,195,76,48,196,
32,163,072
690 DATA196 173,45,196 162,6
6 160,00 1067
700 DATA32 120,196,173,46,16
6,162,00,992
710 DATA160,89,32,120,100,17
3,47 1068 1012
720 DATA162,70,160,00,32,120
,195 35 801
730 DATA72 134 69 132,70,169
,255,72,973
740 DATA32,231,175,104,104,1
0,10,73,746
750 DATA255,160,200,32,162,1
79,76,155,1227
760 DATA195,160,32,162,179,1
65,102,73 1076
770 DATA255,133,102,165,71,1
64 72,32 995
780 DATA212 187 96,120,173,0
,220,72,1080
790 DATA173,2,220,72,169,15
141 2 795
800 DATA220,173,0,220,41,239
,141 0,1034
810 DATA220,160,8,32,38,156,
173,0,827
820 DATA220,10,10,10,10,141,
45,196,642
830 DATA173,0,220,9,16,141,0
,220,779
840 DATA160 5,32,38 156,173,
0,220,824
850 DATA41 15 13,45,156,141,
45,196,682
860 DATA173,0,220,41,239,141
,0,220 1034
870 DATA160,5,32,38,156,173,
0,220,824
880 DATA10,10,10,10,141,46,1
96 173 536
890 DATA0,220,9,15,141,0,220
,160,756
900 DATA5,32,38,156,173,0,22
0,41 705
910 DATA15,13,45,100,141,46,
196,173,826
920 DATA25,212,201 255,208 1
,44,169 1115
930 DATA0 141,47,196,104,141
,2,220,651
940 DATA104,141 0,220,88,66,
234 234 1117
950 DATA234,135,208,250,66,0
 0 0,924
960 DATA120 169 127,141,13,2
20,141 13 944
970 DATA221,160,255,141,2,22
0 169 127,1304
980 DATA141 0,220,169,6,141,
14,220 913
990 DATA141,15,220,141,14,22
1,141,15,900
1000 DATA221,00,96,-1
```

Sticky Cursors

Here's an interrupt which gives a superior method of cursor control

This is a utility to allow a joystick, plugged into Port 2, to emulate the cursor keys. The program is interrupt-driven and resides in an unused area of memory. It works by checking the status of port 2 every so often and when it finds that the joystick is not centred, the appropriate control code is inserted into the keyboard buffer queue.

The joystick will perform the following functions:

POSITION	FUNCTION
Centred	Nothing
Up	Move cursor up a line at a time
Down	Move cursor down a line at a time
Left	Move cursor left (with wrap around)
Right	Move cursor right (with wrap around)
Fire and up	Cancel insert and quote mode (C128 only)
Fire and down	Clear from cursor to end of screen (C128 only)
Fire and left	Clear from cursor to start of line (C128 only)
Fire and right	Clear from cursor to end of line (C128 only)

```
PROGRAM  STICKY CURSOR - C128

10 FOR P = 4864 TO 5122
20    READ X$
30    POKE P,DEC(X$)
40 NEXT P
1000 DATA 78,A9,0D,8D 14,03,A9,1
3,8D 15 03,58,60,CE A1,13,F0,03,
4C,85 FA
1010 DATA A9,03 8D,A1 13,A0,00 D
C,8D,A2,13 29,10,D0,34,AD,A2,13,
29 01,D0
1020 DATA 05,A9,4F 38 4C 8C 13,A
D,A2 13 29,02,D0,05,A9,40,38,4C,
8C 13 AD
1030 DATA A2 13,29,04 00 D0,A9,5
0 38 4C 8C 13,AD,A2,13,29,08 D0
C0,A9 51
1040 DATA 38,4C,8C 13 AD A2 13,2
9,01,D0,D8,A9,91 10,4C 8C 13,AD
A2,13,29
1050 DATA 02,D0,06,A9,11,10,4C,8
C,13,AD A2,13,29 04,00,D8,A9,90,
10 4C,8C
1060 DATA 13,AD,A2,13,29,00,D0,0
C,A9,10 10,4C 8C,13,90 04,A2,10,
D6,F0 8D
1070 DATA 4A,03,A5,00,D0 02,E6,0
0,4C,65,FA,03,03,FF 01,7F,20,D2,
FF,4C 45
1080 DATA 14,20,4E DA,A9,AA,A0,1
E,20,71,09,20,82,09,20,5D,DA,09,
80 48,AD
1090 DATA 14,20,F0,03,20 38,10,2
0,F8 09 68,4C,E7,DA,38,A5,39,E0,
08 20 85
1100 DATA 38,A5 3A,E0 09,20,D5,3
8,F0 04,A9,09,85,0C,20 4E DA A9,
00,A0 1F
1110 DATA 20 71,09,20,1C,13,A5 0
C,C9,05,F0,03,20,37,09,A9,00,A6,
39 A4,3A
1120 DATA 20,D5,FF,90,4F,44,30,0
D
```

```
PROGRAM  STICKY CURSOR - C64

DB   10 FOR I = 40854 TO 40959
F2   20 READ A
B6   30 POKE I,A
9E   40 NEXT I
71   50 SYS 40854
25   100 DATA 120,169,171,141,20,
     3,169,158
AE   110 DATA 141,21,3,88 169 149
     133,55
58   120 DATA 169,159 133,56 96,2
     06 254,158
00   130 DATA 240,3,76,49,234,169
     3,141
55   140 DATA 254,158,173,0,220,1
     41,255,158
B4   150 DATA 41,16,208,3,76,49,2
     34,173
21   160 DATA 255,158,41,1,208,9,
     169,149
B2   170 DATA 76,242,158,173,255,
     158,41,2
75   180 DATA 208,9,169,17,76,242
     ,158,173
E3   190 DATA 255,158,41,4,208,5,
     169 157
DE   200 DATA 78,242,158,173,255,
     158,41,8
C7   210 DATA 208,192,169,23,141,
     119,2 165
C7   220 DATA 198,208,2,230,198,7
     6,49,234
3A   230 DATA 1,127
```

Installing the C128 interrupt

Type in the Basic program (C128 version), save and then run it. The program will then be installed. Save the machine-code from 4864 to 5122 as a binary file.

Type SYS 4864 to run the program and the joystick will emulate the cursor keys. In future when you need the utility it can be loaded directly into memory from the binary file and run by typing SYS 4864.

Installing the C64 interrupt

Type in the Basic program (C64 version), save and run it. The program will then be installed. Type SYS 40854 to run the program and the joystick will emulate the cursor keys. The extra functions available on the C128 version are not catered for because they are not included in the operating system. The program resides from 40854 to 40959, so the top of Basic memory will be lowered when the program runs.

by S Grewal

64 Tips for the 64

by Eric Doyle

A host of useful suggestions to improve your programming prowess

The Commodore 64 harbours many secrets deep inside its memory banks Here are 64 of the best but delve around and you may find many more All the hints were revealed by poking and peeking around the memory Some are old, some are new but all will improve programs and programming beyond your wildest dreams

Although these tips are principally for the Commodore 64, the technical section will help you to convert many of them for the C128

1 High and low bytes in decimal

The low byte is derived from the high byte calculation

HIBYTE = INT(location/256)
LOBYTE = location - (HIBYTE * 256)

2 DEC to HEX conversion

```
10 INPUT"NUMBER IN DECIMAL",DE
C IFDEC>65535THENPRINT"TOO BIG
" GOTO10
20 F=4096 FORA=1TO4
30 D%=DEC/F DEC=DEC-D%*F
40 D%=D%+48 IFD%>57THEND%=D%+7
50 HX$=HX$+CHR$(D%) F=F/16
60 NEXT
70 PRINT"$"HX$
```

3 HEX to DEC conversion

```
10 INPUT"NUMBER IN HEX",HX$
20 L=LEN(HX$)
30 IF LEFT$(HX$,1)="$"THENHX$=
RIGHT$(HX$,L-1) L=L-1
40 IFL>4THEN 10
50 IFL<4THEN HX$="0"+HX$ L=L+1
 GOTO50
60 FORA=1TO4 D=ASC(MID$(HX$,A,
1))-48
70 IFD>9THEND=D-7 IF D>15THENP
RINT"NUMBER ERROR" GOTO10
80 DEC=DEC*16+D NEXT
90 PRINTDEC
```

4 Using ROM routines

Locations 780 to 783 represent the A, X, Y and status registers respectively By poking suitable values to the relevant location, the registers can be set before ROM routines are called

5 Double byte HEX to DEC

```
10 INPUT LOW BYTE LOCATION IN
DECIMAL",LOC
20 POKE781,PEEK(LOC) REM LD BY
TE INTO X REGISTER
30 POKE780,PEEK(LOC+1) REM HIG
H BYTE IN A REGISTER
40 SYS48589 REM PRINT A & X AS
 DECIMAL IN ASCII
```

6 Sub routine 48589

This is where the LIST routine goes to convert line numbers to decimal It then prints the value to the screen Use it for datamaker programs or for any routine where a decimal number has to be printed in ASCII characters The A-register carries the high byte value and the X-register carries the low byte

7 Simulated PRINT@ command

```
10 PRINT"[CLR]"
20 ROW=12:COLUMN=15
30 POKE 214,ROW
40 POKE 211,COLUMN
50 SYS58732
60 PRINT"HI THERE!"
```

8 Alternative PRINT@ command

```
10 PRINT"[CLR]"
20 ROW=12 COLUMN=15
30 POKE 781 ROW  REM X REG
40 POKE 782 COLUMN REM Y REG
45 POKE 783,PEEK(783)AND 254
50 SYS59634
60 PRINT"HI THERE!"
```

9 Finding the cursor
A similar routine can also locate the cursor

```
10 PRINT"[CLR,DOWN6,RIGHT6]";
REM 6 ACROSS AND 6 DOWN
20 POKE 783,PEEK(783)OR1
30 SYS58634
40 ROW=PEEK(781)
50 COLUMN=PEEK(782)
60 PRINTROW,COLUMN
```

10 Code space
Machine code routines are faster if they access zero page locations but the Basic operating system leaves very few possibilities Locations $02 and $FA to $FE (250–254) are normally unused but take care when using cartridges because they sometimes use these bytes

Location $FF can also be used but it is best to use this for transient values

Also in low memory, $02A7 to $02FF (679–767 can be used for small coded programs and disk users can access the cassette buffer at $0334 to $03FF (820–1023) Cassette users can only use $0334 to $033B (820–827) and $03FC to $03FF (1020–1023)

$C000 to $CFFF (49152–53247) is the coder's paradise, four kilobytes of protected memory which doesn't exist as far as the C64's operating system is concerned

11 Room at the top
Extra protected space can be created by lowering the top of memory

POKE 55,0 POKE 56,128 CLR

Location 56 holds the high byte and 55 takes the low byte value In the example this gives decimal 32768 (hex $8000)

12 Raising Basic
In a similar way the bottom of Basic can be raised–

POKE 43 1 POKE 44,16 POKE 4096 0 CLR

The location is the new start of Basic+1 or 4097 ($1001) Location 4096 ($1000) must contain a zero otherwise a syntax error will occur when the RUN command is used

13 Memory SAVE
Once a machine code program has been poked into memory this routine will save it

```
10 REM FILENAME
15 FL=0 LS=0 HS=12*16 LE=6 HE=
12*16
20 POKE 183,FL
30 POKE 187,PEEK(43)+6 POKE188
,PEEK(44)
40 POKE193,LS POKE194,HS
50 POKE174 LE POKE175,HE+1
60 POKE186,8 REM 8 FOR DISK
70 POKE185,0 REM SECONDARY ADD
RESS
80 SYS62854
```

The actual filename for the save should be stored where 'filename' appears and FL is the number of characters in the name LS HS, LE and HE are the high and low bytes of the code block's start and end locations

14 Adding programs together
Frequently used subroutines can be added to a program in memory First of all, PEEK locations 43 and 44, noting down the values (usually one and eight) POKE 43, PEEK(45), and POKE 44, PEEK(46), to set the start of Basic to the end of the program in memory and then enter NEW Next, load the subroutine as normal using D (rather than ,D,1), where D is the device number Finally, the original values can be poked back into 43 and 44

One word of warning the line numbers of the appended subroutine must start at a higher value than the program in memory Failing this, use a renumber routine on the program taking care with GOSUB, GOTO and ON commands

15 Directory tricks
LOAD "$$",8 only loads the header and 'free blocks' lines

LOAD "$A*",8 will only load programs starting with the letter before the asterisk

LOAD "$*=S" will load only the SEQ files Similarly with P, U, R substituted for S, PRG, USR or REL files can be selected

16 Scratching around
To scratch all the files on a disk within a particular category (PRG, SEQ, REL, USR) use the form

OPEN 1,8,15,"SO *$=P"
CLOSE1

17 Which device?
To automatically sense, from within a program the device which is currently in use DEV = PEEK(186) will store the number of the last device used as a variable for use in file operations

18 Closing files
To make sure all files are closed use the CLALL call in the Kernal with SYS 65511 This closes all open files but be careful if CMD has been used Sometimes a mere CLOSE command leaves the screen editor in a confused state To exit safely use the format

PRINT#4 CLOSE4

19 Selecting a bank
To point the VIC chip at a different block of memory

POKE 56578, PEEK(56578) OR 3

POKE 56576, (PEEK(56576) AND 252) OR a

Where 'a' is 3 for the normal (default) bank from the start of memory to 16383, 2 for 16384 to 32767 1 for 32768 to 49151 or 0 for 49152 upwards

20 Moving the screen
Relocate the screen position within a bank with

POKE 53272, (PEEK(53272) AND 15) OR b

Where 'b' takes a value from Table 1 and the actual location is the bank location plus the 'b' value

TABLE 1 - Bank location values

b	Start Location	b	Start Location
0	0	128	6192
16	1024	144	9216
32	2048	160	10240
48	3072	176	11264
64	4096	192	12288
80	5120	208	13312
96	6144	224	14336
112	7160	240	15360

21 Movlag the characters
To relocate the 2K area from which RAM character information is taken
POKE 53272, (PEEK(53272) AND 240) OR c

Where 'c' bas a value from Table 2

TABLE 2 - Character relocation table

c	Start Location
0	0
2	2048
4	4096
6	6144
8	6192
10	10240
12	12288
14	14336

22 Relocating character data

```
10 POKE 56334,PEEK(56334)AND 2
54
20 POKE1,PEEK(1)AND251
30 FORA=8TO2047
40 POKE NEWLOC+A,PEEK(53248)+A

50 POKE1,PEEK(1)OR4
60 POKE56334,PEEK(56334)OR1
```

This routine cannot be stopped because the keyboard is disabled in line 10 Substitute a value derived from Table 2 for 'newloc'

23 GET with cursor
10 GET A$ POKE 204,0 IF A$=""THEN 10

24 Display controls in PRINT
POKE 212,1 PRINT"[CLR] HELLO"

25 Best hi-res location
Set the VIC chip to Bank 4 and the screen to location 57344 ($E000) The ROM doesn't have to be switched out because the screen 'looks' through it A routine is needed to switch out the ROM if the screen needs to be PEEKed but POKEs can be performed with the ROM in place

26 Easy INPUT
10 INPUT"DO YOU WANT TO SAY YES[RIGHT2]Y [LEFT3]" A$

27 Position in colour RAM
PRINT PEEK(243) + 256 * PEEK(224)

28 Position in screen RAM
PRINT PEEK(209) + 256 * PEEK(210) + PEEK(211)

29 Switch screen off/on
Off POKE 53265, PEEK(53265) AND 239
On POKE 53265, PEEK(53265) OR 16

30 Supervision
When using screen routines use base addresses for the screen and colour RAM, CO=55296:SC=1024 This means that a POKE to the video can be colour co-ordinated

POKE SC+50,character:POKE CO+50,colour

31 Key detection
PEEK location 653 to see if the SHIFT (value 1) CBM (2) or CTRL (4) keys have been pressed If two keys are held down at the same time the value found in 653 is the sum of the key values For example, a value of five means that the CBM and CTRL keys are down

32 Clearing the key buffer
Before detecting a keypress, make sure that the key buffer is empty by POKE 198,0

33 Filling the key buffer
Loading from inside a program has its drawbacks A better way is to place the loading sequence in the key buffer so that the next program loads as though the commands have been typed in

The buffer queue is located at 631 and continues through the following nine bytes. This means that only ten characters can be stored here as ASCII codes A routine such as this would suffice

```
10 PRINT"[CLR][5D]"+CHR$(34)+
"ZULU"+CHR$(34)+":" 0,1[DOWN43]:
PRINT"RUN"
20 POKE631,19 POKE632,13 POKE6
33,13
30 POKE198,3
```

This would appear at the end of the first program and would load a program called ZULU To save on buffer space, the commands are written to a cleared screen They have to be correctly spaced to allow for the on-screen messages that the LOAD routine displays

The buffer is poked with a HOME command to place the cursor on the top screen line and this is followed by two returns Location 198 has to be told that these three characters are waiting in the buffer

When the program ends the first return automatically executes the LOAD command and then the loaded program automatically runs

A modified version of this could load a series of machine code routines and execute a SYS command at the end. With a ten character buffer there could be 9 returns stored that means nine commands executed after the first program ends

34 List protect 1
To protect a program from prying eyes use POKE 775, 200 This prevents listing and can be restored by poking the original value of 167 back again

35 List protect 2
Another protection method is to use a REM statement which contains a shifted L When the program is listed it produces a syntax error after the statement This is easily overridden by listing the program and pressing the return key on the listed line Using this method in conjunction with the next method can improve this command

36 List protect 3
A REM statement with a few delete symbols (reversed T) can create havoc with the LIST command

To insert a delete symbol, type REM and then two pairs of inverted commas delete the second pair of quotes After the remaining set of quotes press the insert key (shifted INST/DEL key) once for each character of the current line which you want to hide Next press the delete key the same number of times When the program is LISTed and deletes will each erase a character which lies to their left For example

```
10 SYS49152 REM"[DEL13]"
   REM[SL]
```

would pull the second REM over the SYS command making the line look and behave like a REM with a shifted L

37 Key repeat

The only keys which will repeat when held down are the cursor keys and spacebar Poking a value of 128 to location 650 makes all keys repeat but poking 64 instead will prevent any of the keys from repeating

38 Repeat delay

If all keys are set to repeat, the gap between the first character appearing and the row of repeats can be altered by poking different values to location 651 The maximum delay is given when the poked value is 255, causing a delay of around four seconds

39 UnNEW

Accidentally NEWing a program before it is saved does not mean that the program is lost forever The easiest way to recover is to reset the line link vectors, rechain the program and restore the end of program pointer

One way to do this is to store a program at the top of memory Enter the following line

```
POKE 43 1 POKE
44 159 POKE 46 159 POKE
40704 0 NEW
```

This sets the start of Basic at 40704 ($9F00). The following UNNEW program can not be typed in and saved with ",dev 1" where dev is 1 or 8 for tape or disk

```
10 FOR A=2050TO40703
20 GOSUB50
30 NEXT:FA=40704:THENEND
35 A=INT(B/256):B=B-A*256:?
40 POKE251,B:POKE252,A
50 POKE44,B:POKE2050,255:SYS942
91:POKE45,PEEK(251):POKE46,PE
EK(252):CLR:END
60 IFPEEK(A)=0ANDPEEK(A+1)=0AN
DPEEK(A+2)=0THENB=A:A=49999
70 RETURN
```

The program will always load at 40704 as long as the ',1' is added

When a program is to be restored after NEW enter the line of pokes mentioned at the beginning of this section and load the UNNEW program

The program can now be RUN but it may take a while before the READY prompt appears again. When it does, the program will have been restored

40 Extending Basic

When a Basic program is decoded by the operating system it calls in the Basic lines for analysis in the input buffer starting at $0200 (512) Part of the decoding routine lies in RAM and this is where an extra routine can be wedged in

This routine is stored at $0079 and looks like this

```
$0073 INC $7A
$0075 BNE $0079
$0077 INC $7B
$0079 LDA $0201
$007C CMP #$3A
$007E BCS $008A
$0080 CMP #$20
$0082 BEQ $0073
$0084 SEC
$0085 SBC #$30
$0087 SEC
$0088 SBC #$D0
$008A RTS
```

A suitable break point can be created by moving CMP #$20 to $007C and BEQ $0073 to $007E Location $0080 can now point to the new decoding routine (eg JMP $C000)

The routine grabs a byte from the buffer and attempts to decode it By using a prefix on all of the new commands, decoding becomes easier The 'at' symbol located next to the asterisk on the keyboard is a good prefix so the new routine could look like this

```
$C000 CMP #$40
$C002 BEQ $C012
$C004 CMP #$38
$C006 BCC $C00B
$C008 JMP $C011
$C00B SEC
$C00C SBC #$30
$C00E SEC
$C00F SBC #$D0
$C011 RTS
$C102 DECODE ROUTINE
```

The decode routine can call extra bytes from the buffer with JSR $0073 until a colon is detected After execution of the command the new routine should hand back control to the normal operating system with JMP $0073

41 Setting up interrupts

Once written, an interrupt routine can be called by

```
SEI
LDA #$1F
STA $DC0D
STA $DD0D
LDA $DC0D
LDA $DD0D
LDA #$low byte
STA $0314
LDA #$high byte
STA $0315
LDA #$01
STA $D01A
CLI
RTS
```

The high and low bytes refer to the location of the new interrupt routine Somewhere in the new routine three extra lines should be included The first two are

```
LDA #$01
STA $D019
```

This can be located anywhere in the new code but the third extra line should be used instead of RTS

```
JMP $EA31
```

This checks the keyboard to see if an input has occurred If the keyboard check isn't necessary the routine should end with

```
PLA
TAY
PLA
TAX
PLA
RTI
```

42 WAIT problems

The WAIT command is probably the least used and most misunderstood Basic term Its use is princi-

pally for I/O actions because it has to use a memory location that can be changed externally

The most common applications are to test for a datasette or keyboard keypress

WAIT 1,32,32
WAIT 197,8

The first key is the datasette sensor which waits until bit 5 of location 1 is set by pressing any of the motor keys

The second command peeks into the keypress register Any key decode value which sets bit 3 will terminate the pause so 'A' (value 10) will be accepted but 'R' (17) would have no effect

Another use is to detect the pressing of the CTRL, CBM or SHIFT keys through location 653 WAIT 653 1 will detect the SHIFT key, WAIT 653,2 finds the CBM key and WAIT 653 4 is the CTRL command Alternatively, WAIT 653,6 will detect either the CTRL or CBM keys

43 Wait a jiffy
One internal use of WAIT is to create a time delay when used with the TI$ register

TI$="000000" WAIT 161 1 will give a 4 second pause

44 Unofficial pause
The pause routine called by the tape loading routines can be used as an alternative to the usual 'press any key' pause SYS 58592 followed by POKE198 0 will give an infinite pause and forget which key was pressed The key must be one of those to the left of the keyboard

45 Colour change
Character colours can be changed through control codes in PRINT statements but a better way for certain functions is to poke the colour value direct to location 646

46 Flash data
It can be very boring waiting for a program to read in data Sometimes it can take so long that you start to wonder if the computer has hung up One way to reassure the user is to add an extra POKE command to flash the border The nice thing about this location is that it will accept any value up to 255 without a murmur even though there are only sixteen colours to choose from

For numerical data POKE 53280,A will create a border flash if A is the data value which has just been read in If the data is greater than 255, a small division routine to keep the numbers down to acceptable values can be included

With string data a modified version can be used taking ASC(A$) as the value

47 Screen shake
One less serious function of location 53270 can be shown with this line

```
FOR A = 0 TO 255:POKE
53270,A:NEXT
```

The screen shakes as though a violent earthquake was stirring inside the 64 For added effect, poking A to 53281 will really blow your mind

48 Reset prevention
The RUN/STOP with RESTORE reset can be prevented by POKE 808 225 The function can be reenabled by POKE 808,237

49 Cold restart
When a program has been altering the start of Basic and various other parameters, the easiest way to get the C64 back to normal is by using SYS 64738 instead of the END command

50 Scrolling screens
An upwards screen scroll can be forced at any time by SYS 59626

51 Making spaces
SYS 59749 will open up a screen line beneath the current cursor position effectively scrolling the screen downwards

Combining this routine with the normal scroll a routine can be devised to give quite stunning effects

```
10 PRINT"{HOME,YELLOW,SF4@,C7}
" POKE214,255
20 FORA=0TO23:SYS59749:NEXT
30 FORA=0TO1200:NEXT
40 FORA=0TO25:SYS59626:NEXT
50 FORA=1024 TO2023:POKEA,A-IN
T(A/256)*256:NEXT
60 GOTO10
```

52 Screen flipping
Location 648 directs the operating system to the current screen area

```
10 PRINT"{CLR}THIS IS SCREEN 1

20 PRINT"PRESS ANY KEY"
30 WAIT198,1
40 POKE56579,PEEK(56579)OR3
50 GOSUB160
60 PRINT"{CLR}THIS IS SCREEN 2
{DOWN3}"
70 INPUT"TYPE ANYTHING IN",A$
80 GOSUB200
90 PRINT"{HOME,DOWN3}@ACK AGAI
N"
100 PRINTA$
110 FORA=1TO10
120 FORA=0TO1000:NEXT:GOSUB180

130 FORA=1TO1000:NEXT
140 GOSUB200
150 NEXT:END
160 POKE648,132
170 POKE56576,(PEEK(56576)AND2
52)OR1
180 POKE53272,(PEEK(53272)AND1
5)OR16
190 RETURN
200 POKE648,4
210 POKE56576,(PEEK(56576)AND2
52)OR3
220 RETURN
```

53 Open files
With complex programs, it's best to keep track of the number of open files PEEK(152) will reveal the current number of active files

54 Motor kill
To stop the datasette motor at any time, SYS $FCCA

55 String grabbing
Strings can be pulled out of memory by using the ROM routine at 43806 The string must end with a quotation mark or a zero and the start address is stored in the accumulator (780) and Y register (782) in low byte/high byte format

```
10 REM "WHO STOLE THE @YTE@? "
:.
20 POKE780,115:POKE782,220:SYS
43806
30 POKE780,96:POKE782,220:SY@4
3806
40 PRINT PRINT
50 POKE780,0:POKE782,0:SY@4380
6
```

56 Last filename
The name of the last file to be loaded into the computer can be found from a ROM routine which the

computer uses to print the 'searching for' message SYS 62913.

57 Faster Basic
Maintaining the screen is a time consuming job for the 64. Complex calculations can be speeded up by five percent if the screen is blanked out first

POKE 56325, PEEK(56325) AND 239

To get the screen back again the command becomes

POKE 56325, PEEK(56325) OR 16

58 Memory configuration
Switching ROMs in and out of the 64 memory is controlled by the first three bits of location 1.

Bit 0 controls the Basic ROM at 40960 ($A000)

Bit 1 switches the Kernal ROM at 57344 ($E000)

Bit 2 is responsible for the character ROM at 53248 ($D000)

59 One-line clear
A partial screen clear can be achieved by using part of the in-built screen clear command. The row to be cleared is poked into the X register (780). Remember that the numbering starts with line 0

```
10 FOR A=1024 TO 2023
   POKE A, 1 NEXT
20 FOR A=5 TO 20 POKE
   781, A
30 SYS 59903
40 NEXT
```

60 Line snatching
The screen routines can also be used for line grabbing.

Location 59855 is the start of a routine which will copy a line from anywhere in memory and place it on the screen. The only thing to remember is that the data must be written in screen poke values and be 40 characters long.

The low/high byte information for the start of the line is poked into locations 172/173

POKE 172, 0 POKE 173, 8 SYS 59855

The line is printed at the current cursor position which should be set before the routine is called (see tip 7)

Used with imagination, this routine can be used to create special effects such as screen rotation or a one line scroll

61 Wot, no query
Sometimes a question mark seems odd when an input is required. Not all inputs are questions so why should there be a question mark?

POKE 19,1 before using INPUT will cure the problem

62 Tape headers
The cassette buffer is a misnomer, it is actually the tape header buffer. When the header is FOUND switch off the datasette and press RUN/STOP. Run the following program to find out what's in the buffer

```
10 CB=828
20 S=PEEK(CB)
30 LA=PEEK(CB+1)+PEEK(CB+2)*256
40 EA=PEEK(CB+3)+PEEK(CB+4)*256
50 FORA=CB+STOCB+20
60 FS=FS+CHR$(PEEK(A)) NEXT
70 FORA=CB+21TO1819 IFPEEK(A)=20THEN90
80 NC=1
90 NEXT
100 PRINT"{CLR,DOWN,5PC15}TAPE INFO"
110 PRINT"{5PC15,5ES}"
120 PRINT"{DOWN}SECONDARY ADDRESS =",S
130 PRINT"{DOWN}START ADDRESS{5PC5}=",LA
140 PRINT"{DOWN}END ADDRESS{SPC7}=",EA
150 PRINT"{DOWN}FILENAME{SPC10}=";POKE212,1 PRINTFS
160 IFMCTHENPRINT"{DOWN}MACHINE CODE IN RESIDENCE"
```

63 Controlled Print
It's no secret that control codes can be used for changing the print colour but have you tried non-colour control characters. After typing PRINT, open quotes, hold down CTRL and a letter and see what happens when the program runs.

CTRL M can't be used because it does an automatic shifted return and erases the rest of the line that you're trying to enter. CTRL N automatically switches the computer into lower case

64 FAC facts
Floating point is difficult to understand but can easily be used by machine code routines

The conversion of the numbers is performed by the routine at $B391. It takes a number stored in the Y/A registers in low/high byte format and stores it in a special register known as FAC#1 (Floating point ACcumulator). If two numbers are to be operated on they must be transferred in the correct order to FAC#1 and FAC#2.

To transfer a number from FAC#1 to FAC#2 the routine at $BC0F is used. This is shown in the example routine which divides 1000 by 4 in hex

```
LDA #$03
LDY #$E8
JSR $B391
JSR $BC0F
LDA #$00
LDY #$04
JSR $B391
JSR $BB12
JSR $BDDD
LDY #$05
*LDA $0100,Y
STA $00F9,Y
DEY
BNE *
RTS
```

To convert the result in FAC#1 into a usable form, $BDDD returns the result in ASCII to $0100 from which it can be stored and used in this case from location $FA

Other useful routines are available

$BBFC	transfer FAC#2 to FAC#1
$B86F	add FAC#1 to FAC#2
$B853	subtract FAC#1 from FAC#2 the result is stored in FAC#1
$BA30	multiply FAC#1 by FAC#2 and store the result in FAC#1
$BB12	divide FAC#2 by FAC#1 and store the result in FAC#1
$BF7B	raise FAC#1 to the power FAC#2 and store the result in FAC#1
$BF71	calculate the square root of FAC#1 and store the result in FAC#1

If you discover an interesting routine for the Commodore 64, why not let everyone in on the secret and send it to Tips for the Serious User, Your Commodore, ASP Ltd, 1 Golden Square, London W1R 3AB

LIFESAVERS 3	C128	TEXT SCREEN DUMP	1/1

This utility will dump tha current text screen from inside a Basic program. It raads the location where tha screen is storad and translates them into ASCII code valuas ready for printing.

Place the program inside a listing and the command GOTO 10 will set tha process in motion.

```
10 REM TEXT SCREEN DUMP
20 REM BY ANDREW GORRIE
30 FAST : P$=CNR$(16) : A$="" :
OPEN4,4,7
40 AA=1024:FOR P=0 TO 24
50 FOR N=AA TO AA+39 : GOSUB 60 :
NEXT : AA=AA+40 : PRINT#4,P$ "15"
A$: NEXT P :SLOW : END
60 IF PEEK(N)>63 AND PEEK(N)<91
THEN Q=32
70 IF PEEK(N)>128 THEN Q=-128
80 IF PEEK(N)<26 TNEN Q=64
90 A$=A$+CNR$(PEEK(N)+Q) : Q=0 :
RETURN
```

LIFESAVERS 4	C64	RESTORE BORDER CHANGE	1/1

Running this program means that the border can be changed by pressing the RESTORE key. The colour cycles through to a new value for each press of the key.

Tha program works by resatting the STOP routine vector at 808-9.By altering thesa you can point the restore key to any routine you like. The border change routine has been chosen to highlight a side effact of this mathod. When a program is loaded or saved,the border changes as the ROM routinas call on the STOP routine while performing thair tasks.

Pressing RUN/STOP with restore will deactivate the border routine and POKE 808,0:POKE 809,192 will activate it.

```
10 REM RESTORE KEY BORDER CHANGE
20 C=0:FOR I=0 TO 5: READ A:POKE
49152+I,A:C=C+A:NEXT
30 IF C<>1037 TNEN PRINT"ERROR IN
LINE 40":END
40 DATA 238,32,208,76,237,246
50 POKE 808,0:POKE 809,192
60 PRINT"PRESS RESTORE"
```

A Bit More

Turn your C64 into a hi-res machine with a bitmap, 16 sprites, a redefinable character set, 8K of storage plus 21 commands – but retaining 36K for Basic!

by Malcolm W Gallon

This utility combines two ideas in the one program Firstly, the memory map is reconfigured so that the screen, character set, and bitmap are in Bank 3 (from 49152 to 65535) and, secondly, an arsenal of 21 machine-code commands is loaded into the area of memory once used by the screen (from 1020 to 3794), the start of Basic being moved up to 3795 to protect it. If you have a look at the memory map provided you will see that almost the whole of the memory is now usable The exceptions being the I/O area from 53248 to 56344 and the area from 0 to 3494, which still leaves some 57945 bytes free

The program is in two parts The first, called MCLOAD, is stored from 680 to 762 and forms a machine-code loader which sets the start of Basic memory to 3795 and then loads in the main program, called MCFILE, and calls the reconfigure command (SYS1020), finally ending with NEW to set the Basic pointers correctly

MCFILE is the program containing the machine-code commands and consists of a jump table (from 1020 to 1083) for the command routines, followed by the routines themselves between 1086 and 3794

The following is a detailed list of all the new commands available

RECONFIGURE – SYS1020

This is the command that sets up the new memory configuration and can be used at any time The screen, sprite pointers, sprites and bitmap maintain their relative positions but move up into Bank 3

The screen is at 1024+49152=50176, the first sprite pointer is at 2040+49152=51192, sprites 0–15 are at 0*64+49152=49152 to 15*64+49152=50112 and the bitmap is at 8192+49152=57344

The character set is now in RAM at 51200 to 53248 and consists of the first 256 characters of the normal set but it can be redefined at any time

RASTER ON – SYS1023

The raster interrupt routine is turned on with this command and can be used for both split-screen graphics and sound as detailed later It is not recommended that you leave the raster interrupt routine running when using the LOAD and SAVE or the MEMORY MOVE and MEMORY FILL commands

RASTER OFF – SYS1026

This command is obviously to turn off the interrupt routine

SPLIT-SCREEN – SYS1029, <0 or 1–200>

This command is used in conjunction with the RASTER ON command to set up a split-screen where the top part is bitmapped while the rest is the normal screen display It requires one parameter which sets the split to a screen line between 1 and 200 A value of zero turns the split-screen off

BITMAP ON – SYS1032

This turns on the full screen bitmap If a full bitmap and an interrupt are required at the same time for running the sound routine, then the SPLIT-SCREEN command SYS1029,200 should be used rather than this one If at any time this command does not appear to be working, check that the raster routine has been switched off first because they won't work together

BITMAP OFF – SYS1035

Turns off the bitmap The same rules apply to these interrupts as those detailed in the BITMAP ON command

CLEAR BITMAP – SYS1038

This command clears the complete bitmap If you want to clear only parts of a bitmap see the MEMORY FILL command and Example 3 in the DEMO program

COLOUR BITMAP – SYS1041

The complete bitmap is filled with

the current colour stored in 254, this is set with the SET CURRENT COLOUR command

SET CURRENT COLOUR –
SYS1044,<0–15>,<0–15>

Use this command to set the display colour It requires two parameters, the first is the plot or drawing colour using the standard colour codes and the second is the background colour

PLOT POINT ON BITMAP –
SYS1047,<0–319>,<0–199>,
<0–1–2>

Points on the bitmap screen will be plotted in the current colour Three parameters are required – the first is the X co-ordinate between 0 and 319, the second is the Y co-ordinate between 0 and 199, the third is the mode which can be zero to unplot or erase a point, one to plot a point or two to test if a pixel is on or off If the pixel is off then location 2 will contain a zero if the pixel is on then it will contain the number 100

DRAW LINE –
SYS1050,
<0–319>,<0–199>,<0–319>,
<0–199>,<0–1>

To draw lines on a bitmap, this command can be used with its five parameters The first two are the starting X and Y co-ordinates of the line, the next two are the end of line X and Y co-ordinates, the last is the mode, which can be either zero to erase or one to plot

DRAW CIRCLE –
SYS1053,
<0–319>,<0–199>,<0–129>,
<0–1>

Here is the command which draws circles from four parameters The first two are the X and Y co-ordinates of the centre of the circle, the next is the radius of the circle The smallest circle that can be drawn has a radius of one and the largest has radius 129 The last parameter is once again the mode either zero or one for erasing or plotting

FILL BITMAP –
SYS1056,
<0–319>,<0–199>,<0–1>

This is the command that fills selected areas of the bitmap and it requires three parameters The first two are the X and Y co-ordinates for the starting point of the fill, the third is again the mode of either zero or one The starting point of the fill is important as the routine is not the smartest around Try the following example and you will see what I mean

Set up a bitmap using the following commands in either program or immediate mode

```
SYS1044,2,10          – set colours
                        to red on
                        light red
SYS1038 SYS1041       – clear and
                        colour the
                        bitmap
SYS1032               – turn on
                        the
                        bitmap
SYS1053,160,100,50,1 – draw
                        circle
```

Next try SYS1056,140,100,1 and you will see that the circle is not properly filled Erase the above with SYS1056,140,100 0 and draw the circle again (same as before) and then use SYS1056,160,100,1 The circle will now be filled correctly Have a look at Example 6 in the demo program and you will see some of the limitations of the fill commands. The two fill routines are reasonably fast and compact in code so you can't expect miracles.

FILL WITH CHARACTER –
SYS1059,<0–319>,<0–199>,
<0–1>,<0–255>

This routine works in the same way as the ordinary fill routine but is has an extra parameter This is a character from the character set, numbered 0 to 255, and is used to specify the fill pattern You can fill or erase any area of the screen or even the whole screen with this character and since the character set is redefinable, there are hundreds of possible fill patteros available The limitations of the fill routine also apply to this command

SET SOUND –
SYS1062,<1–3>,
<0–15>,<0–255>,<0–255>,
<0–255>,<0–255>,<0–255>,
<0–15>,<0–255>,<0–255>,

The sound command can be used in two ways and needs ten parameters to work In order, these are the voice, volume, low frequency, high frequency, attack/decay, sustain/release, low pulse, high pulse, waveform and the duration of a note All but one of these operate in the normal way, the exception being the duration

If the interrupt is not turned on, then the duration is simply a set-up command and the particular voice used will have to be gated off in the normal way If, however, the interrupt is on, then the duration parameter comes into effect The length of the note is calculated by

Duration = time in seconds × 50

eg a five second note gives a parameter of 5 × 50 = 250

All three voices can be used at once under interrupt control and you can have split-screen graphics and interrupt controlled sound at the same time

To check if a voice has finished processing, or requires more data, a PEEK to the voice control register is needed For the three voices, the registers are located at 1228, 1129 and 1130 respectively If the Voice 1 register at 1128 contains 100 then the voice is still on If it contains zero then the voice has been gated off The other two voice registers operate in the same way

SET SPRITE –
SYS1065,<0–7>,
<0–1>,<0–255>,<0–15>,
<0–320>,<0–255>,<0–1>,
<0–1>,<0–1>,<0–15>,
<0–15>

This command allows sprites to be set up with only one instruction but it has eleven parameters which need to be specified These are sprite number, sprite off/on, sprite pointer (remember data should be somewhere in bank 3 ie for a pointer set to 13 the data should be at 13*64+49152=49984 onwards), sprite colour, X position, Y position, X expand (0=ordinary, 1=expanded), Y expand, multicolour off/on, multicolour 1, multicolour 2 These all operate in the normal way To turn off any sprite you only need to specify its number followed by zero (eg to turn sprite 7 off use SYS1065 7,0)

MOVE MEMORY –
SYS1068,
<0–65535>,<0–65535>,
<0–32767>

Use of this command will move any

block of memory to any addresss as specified by the three parameters The first is the block's destination address, the second is the starting address of the block to be moved, and the third is its length which is limited to a maximum of 32K.

MOVE MEMORY has access to all of the RAM, including that under the Basic and Kernal ROMS, except for the RAM under the VIC, SID, and the I/O ROMs

As was mentioned earlier it is not advisable to use this command with the interrupt running. If you need to have a split-screen and to move or fill large areas of memory as well, then simply switch off the interrupt before calling this routine and switch it back on immediately afterwards This will cause the screen to flicker or upset sound timings but it is simply a consequence of not being able to have interrupts running while maintaining access to the RAM under ROMs at the same time

FILL MEMORY –
SYS1071,
<0–65535>,<0–32767>,
<0–255>

You can fill any area of memory up to a block size of 32K with a number between 0 and 255 using this command. The parameters required form the starting address of the memory block to be filled, the length of the block (up to 32K), and the number with which the block is filled.

This command uses part of the MOVE MEMORY routine and the interrupt restrictions also apply There are no restrictions concerning which part of memory you are filling so be careful that you don't overwrite something important, such as the operating system and program areas

SAVE MEMORY –
SYS1074,<file name>,<01 or 08–11>,<00>,
<0–65535>,<0–65535>

This is a machine code SAVE routine and can be used either in immediate mode or as part of a program The parameters required are the filename (usual restrictions apply eg name length of 16 characters), device number, the number zero, the starting address of the block to be saved, the end address+1 of the block to be saved

LOAD MEMORY –
SYS1077,
<file name>,<01 or 08–11>,
<00>,<0–65535>

The LOAD command can be used in either immediate mode or within a program and requires the following parameters Firstly the filename, then the device, the number zero (this is essential), and finally the load address.

SET CURSOR POSITION –
SYS1080,<0–24>,<0–39>

This is the last command, and it is used to print text to a specific row and column on the screen, eg SYS1080,10,14 followed by PRINT<text message> will print the message on row 10, column 14

These are all the new commands that are available and if you have a look at the DEMO program provided it should give you some idea of how to make use of them. The best way to find out what can and cannot be achieved is to experiment as much as possible and see what happens

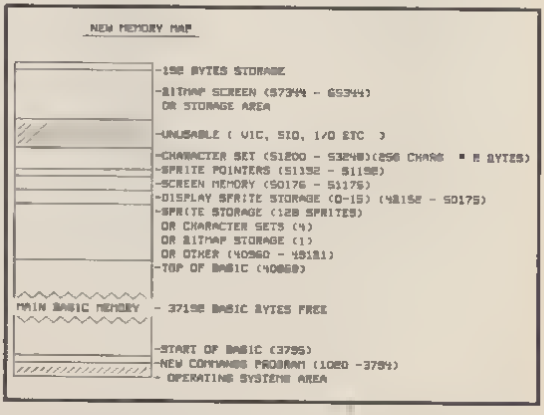
If for any reason you press the RUN/STOP and RESTORE keys, you should put the disk that contains the MCFILE program into the drive, and type SYS680 This will get everything back to normal without losing any Basic program that may be in memory Do not use SYS680 as this contains the NEW command and any Basic program will be lost

Split loyalties

The problems that prevent using the memory move or fill commands with a raster interrupt running relate to the operating system ROMs.

To set the scene, the raster interrupt comes into operation at the specified screen line, either turning on or off the bitmap as necessary The routine then exits via the normal interrupt routine handler at $EA31 to ensure correct Basic operation with keyscan So far so good

The MEMORY MOVE command requires access to the RAM under the Basic and Kernal ROMs, so these are both switched out Interrupts cannot now be allowed to occur since the Basic interpreter is no longer in memory If the MEMORY MOVE command is running, the interrupt flag is set so, if the routine runs for longer than 1/50th of a second (the time it takes



for the raster to scan a complete screen), the raster interrupt will be held up When the MEMORY MOVE command is finished the interrupt flag is cleared and the raster interrupt can now occur Unfortunately, the raster display will be frozen during the MEMORY MOVE interrupt giving a full high resolution screen or a full normal screen. Either way the result is a mess!

The only way round this problem is to avoid using the two commands together or simply switch the raster off before using the MEMORY MOVE command and switch it back on again afterwards using the routines provided

All the above also applies to the MEMORY FILL command because memory has been saved by using some of the subroutines of the MEMORY MOVE command

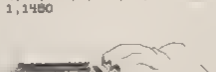
Finally, if you want to load the program without the menu system type LOAD"MCLOAD",8,1 and then SYS680 To get the demo program type LOAD"DEMO",8 then RUN

PROGRAM MCLOAD

```
3B  10 REM MCLOAD
0A  20 REM THIS PROGRAM LOADS AN
    O SAVES IKE MCLOAD FILE
26  30 REM TYPE THIS IN AND SAVE
    IT TO TAPE OR DISK AS "MCLO
    AD BAS"
66  40 REM INSERT YOUR MASTER TA
    PE OR DISK AND RUN MCLOAD BA
    S
67  50 REM THE SCREEN WILL FLASH
     AND THEN THE SAVE WILL OCCU
    R AUTOMATICALLY
06  60 :
7C  70
25  80 BL=4 LN=50 SA=680
CF  90 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15
50  100 READ A CX=CX+A POKE SA+L
    *16+D,A POKES3280,A NEXT D
55  110 READ A IF A><CX THENPRIN
    T"ERROR IN LINE",LN+(L*10) S
    TOP
F8  120 NEXT L READA POKE780,A R
    EADA POKE781,A
CE  130 DATA 32,174,2,76,215,2,1
    63,2,162 8 160,1,32,186,255,
    169,1645
E5  140 DATA 6,162,244,160,2 32,
    189,255,189,0,32,213,255,169
    ,211,133,2232
88  150 DATA 43 169,14,133 44,16
    9,0,141,210,14,32,30,5,169,1
    47,32,1352
2E  160 DATA 210,255,96,169 4,13
    3,186 169,78,141 119 2 169,6
    9,141 120,2073
4D  170 DATA 2,169,87,141,121,2,
    169 13,141,122,2 96 77,67,70
    ,73,1352
F5  180 DATA 76,69,0,0,0,0,0,0 0
    ,0,0,0,0,0,0,145
E9  190 POKE153,169 POKE154,2:PO
    KE174,251 POKE175,2
74  200 POKE187,7 POKE188,8 POKE
    183,6 POKE186,8 REM POKE 186
    ,1 FOR CASSETTE
27  210 POKE185,0 SYS62554
```

PROGRAM MCFILE

```
35  10 REM TYPE IN THIS PROGRAM
    AND SAVE ON A SPARE TAPE OR
    DISK AS "MCFILE BAS"
65  20 REM BEFORE RUNNING PLACE
    THE DISK OR CASSETTE WITH MC
    LOAD INTO THE SAVE DEVICE
81  30 REM IF USING CASSETTE, EN
    SURE IT IS SET TO THE END OF
     MCLOAD
15  40 REM AND MAKE SURE THAT LI
    NE 100 HAS BEEN CHANGED
C6  50 REM RUN MCFILE BAS
51  60 BL=184  :LN=50   :SA=3686
    4
A2  70 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A
65  80 POKE53280,A POKE SA+L*16+
    D,A:NEXT D
81  90 READ A IF A><CX THENPRINT
     ERROR IN LINE";LN+(L*10) ST
    OP
2E  100 NEXT L REM IF USING CASS
    ETTE, INSERT POKE 39715,1 HE
    RE
07  101 SYS38835
50  110 DATA 76,30 5 76 62,4,76,
    68,14,76,203 4,76,147 14,76,
    1007
77  120 DATA 181,14,76,240 4,76,
    6 5,76,123,14 76,144,5,76,16
    3,1263
2F  130 DATA 7 76,111,6,76,124,1
    0 76 117,10,76,186 5 76,252,
    12,1220
EA  140 DATA 76,25 12,76,186,12,
    76,14,14,76 48,14,76,175,14,
    96,550
08  150 DATA 96,96,120,173,14,22
    0,41,254,141,14,220,173,17,2
    0H,41,127,1955
0F  160 DATA 141,17 208 169,0,14
    1,18,208,169,114,141,20,3,16
    9,4,141,1653
36  170 DATA 21 3 173,26,208 9,1
    ,141 26,208,88 96,0,0,0,0,10
    00
84  180 DATA 0,0,0,0,0,0,169,1,4
    4,25,208,240,21,173,113,4,93
    8
C8  190 DATA 201,0,240,3,76,145,
    4,76,174,4,173,25,208 5 1,14
    1,1480
```



```
70  200 DATA 25,208,78,49,234,16
    9 8 44,24,208,208,22,173 17,
    208 5 1692
58  210 DATA 32,141,17,208,169,2
    6,141,24,208,173,112,4,141,1
    8,208,76,1656
9A  220 DATA 134 4,173,17,208,41
    ,223,141,17,208,169,156,141,
    136,2,169,1578
38  230 DATA 10 141,24,208,169,0
    ,141,10,200,32,113,5 76,134,
    4 38,1323
14  240 DATA 91,6,165,20,240,21,
    201,201,144,3,76,72,178,24,1
    65,20 1627
69  250 DATA 105,50,141 112,4 16
    9,100,141,113,4,86,169 0,141
    ,112,4,1461
7A  260 DATA 141,113,4,96,162,32
    ,169,224,133,252,169,0,133,2
    51,160,145 2152
88  270 DATA 251 200,208,251,230
    ,252,202,208,246,96,162,4,16
    9,196,133,252,3060
85  280 DATA 169 0,133,251,168,1
    65 254,145,251,200 208,251,2
    30,252,202,208,3087
02  290 DATA 246 96 120 173,2,22
    1,9,3,141,2,221,173,0,221,41
    ,252,1521
09  300 DATA 3 0,141,0,221,169,1
    8,141 24,208,165,156,141,136
    ,2,169,1744
40  310 DATA 0,133,251,166,200,1
    33,252,165,0,133,253,165,208
    ,133,254 165,2622
15  320 DATA 1 41,251,133,1,160,
    0,177 253,145,251 230,253,23
    0,251 155,2542
A0  330 DATA 251,208,244,230,252
    ,230 254,165,254 201,215,208
    ,234 165,1,9,3122
61  340 DATA 4,133,1 00 96,173,1
    04,4,240,15 173 107 4 208,11
    ,169 1534
C6  350 DATA 0,141,4,212,141,104
    4 76 137,5,208,107,4,173 10
    5,4,1423
00  360 DATA 240,15 173,104 4,20
    8,11,169,0,141,11,212,141,10
    5,4,76,1622
```

```
61  370 DATA 161,5,208,108,4,173
    108 4,240,18,173,109,4,208,
    11,169,1700
54  380 DATA 0,141,18,212,141,10
    5,4,76,185,5,208,108,4 96,38
    ,51,1426
28  390 DATA 6 165,20,240,4,201,
    4,144 3,76,78,178,165,20,168
    136 1602
14  400 DATA 140,111,4,201,3,240
    ,20,201,2,240,8,185,0,141,11
    0,4,1594
1F  410 DATA 76,236,5,165,7,141,
    110,4,76 236,5,169,14,141,11
    0,4 1503
11  420 DATA 32 51,6 32,228,13,1
    41,24,212 38,91,8,32,101,8,1
    74,1221
94  430 DATA 110,4,197,0,212 32,
    91,6,32,101,6,174,110,4,157,
    1,1197
0C  440 DATA 212,32,51,6,32,101,
    6,174,110,4 157,5,212,32,91,
    6,1271
66  450 DATA 32,101,6,174,110,4,
    157 6,212,32,91,8,32,101,8,1
    74 1244
CE  460 DATA 110,4,157,2,212,32,
    91 6 32,228 13,174,110 4,157
    ,3,1335
6A  470 DATA 212,32 51,6,32,101,
    6,174,110,4,157,4 212,32,51,
    6 1270
```

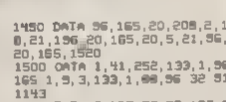
```
50  480 DATA 32,101,6,172,111,4,
    153 107,4,169,100,153,104,4,
    96 32,1348
56  490 DATA 253,174,32,138,173,
    32 247 183 96,165,21,240,3,7
    6,72 178 2083
CF  500 DATA 165 20,56,32,51,6,3
    2,109,14 165 20,133 168,165,
    21,133 1370
34  510 DATA 167 32,51,6,32,97,1
    4,133,165,32,51,6,165,20,240
    ,4,1259
9B  520 DATA 201,130,144,3,76,72
    170,133 82 32 51,6,32,240,1
    3,133,1568
3E  530 DATA 41,165,0,133,80,185
    ,82 133,81,169,1,133,170,185
    ,82,10,1614
84  540 DATA 58,233,1,133,171,58
    ,233,1,133,75,24,165,188,101
    ,81,170,1801
3B  550 DATA 165,167,105,0,168,2
    4,165,169,101,80,176,3,32,14
    1,7,24,1527
B0  560 DATA 185,168,101,80,170,
    185 167 105,0,168,56,165,169
    ,229,81,144 2133
16  570 DATA 3 32,141,7,56,165,1
    68,229,81,170,165,167,233,0,
    168,56 1841
62  580 DATA 169,169,229,80,144,
    3,32,141,7,56,165,168,229,80
    ,170,165,2003
7B  590 DATA 167,233,0,168,24,18
    5,169,101,81,176,3 32,141,7,
    230 80,1777
```

```
B7  600 DATA 230,170,230,170,56,
    165,75,229,170,133,76,176,22
    ,58,169,255,2381
CC  610 DATA 229,75,133,76,230,7
    5,198,81 198,171,198,171,56,
    165,171,229,2455
95  620 DATA 75 133,75,24,165,18
    0,101,80,170,165,167,105,0,1
    68,24,165,1785
E7  630 DATA 169,101,81,176,3,32
    ,141,7,24,165,168,101,81,170
    ,165 167,1761
1F  640 DATA 105,0 168 56,165,18
    9,229,80,144,3,38,141,7,56,1
    65,168,1688
01  650 DATA 229,80,170,165,167,
    233,0,168,56,165,169,229,81,
    144,3,32,2051
BE  660 DATA 141,7,38,165,168 22
    9,81 170 165,167,233,0,168,2
    4,165 169,2108
17  670 DATA 101,80,176,3,32,141
    ,7,165,80,197,81,176,3,76,18
    2,5 1506
4B  680 DATA 56 201,200,176,17,1
    33 253,192 0,240,4,224,64,17
    6,7 134 2117
55  690 DATA 251 138,252,32,180,
    9,58 32,51,6,32 105,14,165,2
    0,133,1554
65  700 DATA 167,165,21,133,168,
    32,51,6 32 57 14,133,169,32
    51,6,1357
D1  710 DATA 32,105,14,165,21 13
    3,76,165,20,133,76,32,51,6,3
    2,57,1201
21  720 DATA 14,133,151,32,51,6,
    32,240,13,133,41,88,165,75 2
    23,167,1578
63  730 DATA 133,170,170,165,76,229,
    168,133,171,176,20,165,170,7
    3,255,133,170,2407
ED  740 DATA 230,170,165,171,73,
    255,133 171 165,170,208,2,23
    0,171 56,165 2535
DA  750 DATA 151,229,168,133,38,
    176,5,73,255,133,38,230,38,1
    65,171 208 2213
22  760 DATA 6,165,170,157,38 14
    4,68,165 76,157,168,144 34,2
    08 6,165 1349
4B  770 DATA 75,157,167,144,26 1
    65,151,157,169,144 11,168,10
    0,133,77,169,2084
EC  780 DATA 0 133 76,76,157,8,1
    69,0,133,77,133,76,76,157,8,
    165,1448
00  790 DATA 151,197,165 144,5,1
    69,100,133,77,133,78,76,157,
    8,165,0,1770
E3  800 DATA 133,77 169 100 133,
    76 76 157,8,165,151,157,169,
    144 34 165 1956
D4  810 DATA 76,197,168 144,15,2
    08,6 165 75,197,167 144,11,1
    69 100,133,1579
0B  820 DATA 39 169,0,133,40,76,
    22,5,169 0 133,39,133,40,76,
    22 1100
22  830 DATA 5,165,76,197,168,14
    4,15 165,75,197,167,144,9,16
    5,100 133 1933
30  840 DATA 39 133,40,76,22,5,1
    69,0,133,39,165,100,133,40 7
    8,22,1200
```

```
D5  850 DATA 5,165,171,74,165,17
    0,108,133,81,169,0,133,82,13
    3,79 133,1803
01  860 DATA 80,32,125,9,165,70,
    208,11,230,167,165,167,208,1
```

```
    5,230,168,2059
6A  870 DATA 76,201,5 198,167,16
    5,167,201,255,208,2,198,168,
    96 165,81,2315
00  880 DATA 229,38,133,81,165,8
    2 233,0 133,82,176,36,165,81
    73 255,1964
3F  890 DATA 133,81,230 81,165,8
    2,73 255,133,82,165,77,208 5
    ,198,169,2137
0E  900 DATA 76,241,8,230 169,56
    ,165,170,229,81,133,81,165,1
    71,229,82,2206
30  910 DATA 133,82 165,80 157 1
    71,208,7,165,79 157,170,208
    1,96 230,2188
08  920 DATA 70,165 79 208,2,230
    ,80,76,173,8,165,38,74,133,8
    1,165,1760
0E  930 DATA 0 133,82,133,79,32,
    128,5,165,40,208,5,230,169,7
    5,47,1534
5C  940 DATA 5,158,169,56,165,81
    ,229,170,133,81,165,82,229 1
    71,133,82,2153
AB  950 DATA 176,52,165,81 73,25
    5,133,81,230,81,165,82,73,25
    5,133 82 2117
D2  960 DATA 165,39,208,13,198,1
    67,165,167,201,255,208,13,19
    8 168,76,101,2342
8E  970 DATA 5 230,167,165;167,2
    08,2,230,168 56 165,38,229,8
    1,133,81,2125
AC  980 DATA 165,82,233,0,133,82
    ,165,79 157,38,208,1,96,230,
    79,76 1864
8C  990 DATA 33,5,165,160,133,25
    2 165 167 133 251,165,169,13
    3,253,32,180 2408
99  1000 DATA 5,96,0 0 32 91,8,3
    2,109,14,165,20,133,251,165,
    21,1144
E8  1010 DATA 133,252,32 51,6,32
    ,97,14 133,253,32 91,6,165,2
    0,201,1558
37  1020 DATA 3,144,3,76,72,178,
    133,41,165,251,41,7,170,56,1
    69,0,1508
CF  1030 DATA 133,80,106 202,16,
    252,133 89,165,251,41,248,13
    3 92,165,253,2369
70  1040 DATA 74,74,74 133,91,74
    ,102,90,74,102,90,101,91,133
    ,91,165 1568
49  1050 DATA 253 41,7 101,90,10
    1,92,133,90 165,51,101,252,1
    05 224 133,1979
D7  1060 DATA 91 120,165,1,41,25
    3,133,1,165,41,240,40,201,1,
```

```
    240,22,1755
25  1070 DATA 160,0,177,80,36,85
    208 7 169 0,133,2,76,109 10
    165,1435
AC  1080 DATA 100,133,2,76,109,1
    0,160 0 177,80,5,89,145,80,3
    2,80,1268
92  1090 DATA 10,76,109,10,160,0
    56,165,255,229 85 133,89,17
    7,80 37 1685
D2  1100 DATA 85 145 80,76,109,1
    0,165 0,133,80,133,81,165,85
    3,41,248,1842
00  1110 DATA 133,82,133 80 10,3
    8,81,10,38 81 24,101,82,133,
    80,165 1331
A5  1120 DATA 81 106,0,133,81,24
    165,0,101 80,133,80,24,169
    198,101,1517
F7  1130 DATA 81 133,81,165,252,
    74,165 251,106 74,74,168 165
    ,254 145,80 2280
0A  1140 DATA 86,165,1,5,2,133,1
    ,88 86,165,100,133 171,76,12
    8,10 1378
78  1150 DATA 165,0,133,171,32,9
    1,6,32,109,14,165,20,133,167
    ,165,21,1428
51  1160 DATA 133 168,32,91,6,32
    87,14 133,165 32,91,6,32,24
    0,13,1285
55  1170 DATA 133,75,201 0,208,7
    165,100 133,76,76,173,10,16
    5,0,133,1653
17  1180 DATA 76,165,171,240,52,
    32,91,6 169,0,133,252,165,21
    240,3,1816
4D  1190 DATA 76,72,170,165,20,1
    0,38,252,10 38 252 10 38,252
    133 251,1795
E4  1200 DATA 24,169,0,101,251,1
    33 251,169,200,101,252,133,2
    52,160 0,177 2373
C2  1210 DATA 251,153,94,0,200,1
    92,8,208 246,165 0 133,170,1
    65,167,133 2283
A0  1220 DATA 251 165,168,133,25
    2,165 169,133,253 169,2,133,
    41,32,155 11 2276
C4  1230 DATA 165,2 167,76,208,2
    0,165,75,133,41,32,172,11,15
    8,251,165,1511
F4  1240 DATA 251,201,255,208,2,
    198 252,76 245,10,165,167,13
    3,251 165,168,2747
84  1250 DATA 133 252,230,251,16
    5,251,208,2,230,252,169,2,13
    3,41 32,155 2550
04  1260 DATA 11,165 2,197,76,20
    0,10,165,75,133,41,32 172,11
    ,76,30,1404
81  1270 DATA 11,165,170,200,34,
    165,167,133,251,165,168,133,
    252,138,253,169,2842
A5  1280 DATA 2,133,41,32 155 11
    ,165,2,197,76,200,3,76,2,11,
    169 1327
9E  1290 DATA 100 133,170,165,16
    8,133,253,165 167,133,251,16
    5,168,133,252,230,2787
53  1300 DATA 253,169 2,133,41,3
    2,155,11 165,2,197,76,208,3,
    76,2,1565
82  1310 DATA 11,96,165,253 201,
    200,176 8,165,252,240,28,201
    ,1,240,18,2255
78  1320 DATA 165,75,240,5,169,1
    00,76,151,11,169,0,133,2,165
    ,100,133,1696
2C  1330 DATA 75,96,165,251,201,
    64,176,232,32,150,5,169,0,13
    3,75,96,1962
4C  1340 DATA 165,171,240,208,32
    ,240,11 165,70 240,0,165,75,
    133,41,32,2002
60  1350 DATA 126,11,96,169,0,13
    3,41,32,126,11,96,32,126,11,
    165,75,1254
FA  1360 DATA 208 4,165 171,208
    1,96,166,75,208,251 32,240,1
    1,165,2,2002
92  1370 DATA 157,70 208,11,165,
    2,201 100 240 236,165,100,13
    3,2,96,165,2107
00  1380 DATA 0,133,2,96 165,253
    ,41,7,170 181 94 133,77,165,
    251,41 1803
76  1390 DATA 7,170,56,169,0,106
    ,202,16 252,133,89,165 77,37
    ,89 201,1765
90  1400 DATA 0,240,5 169 100,13
    3 78 56,165 0 133,78,96,32,9
    1 6,1426
DA  1410 DATA 165,20,72,165,21,7
    2,32 91,6,165,20,72,165,21,7
    2,32 1151
6S  1420 DATA 91,6,160,3,104 153
    ,87,0,136,16,249,165,20,5,21
    ,240,1466
06  1430 DATA 67,165,38,157,100,
    144,65,208 6,165,97,197,99,1
    44,57,216,2025
8C  1440 DATA 24,165,97,101,20,1
    33 97 165,98,101 21,133,96,2
    4,165,99 1541
EC  1450 DATA 101,20,133,99,165,
    100 101 21,133,100,32,170,12
    168,100,136,1663
AF  1460 DATA 98,160,255 177,99,
    145 97,152,208 4,198,100,198
    ,96,136 32,2157
08  1470 DATA 157,12,208,238,32,
    170 12 96 32,170,12,160,0,17
    7,99,145 1728
0E  1480 DATA 87 200,208,4 230,1
    00,230,98,32,157,12 208,240,
    32,170,12,2038
```



```
FB  1490 DATA 96,165,20,208,2,18
    0,21,156 20,165,20,5,21,96,1
    20,165,1520
DA  1500 DATA 1,41,252,133,1,96,
    165 1,9,3,133,1,88,96 32 91
    1143
80  1510 DATA 6,165,20,72,165,21
    ,72 32,91,6,165,20,72,165,21
    ,72 1165
90  1520 DATA 32,91,6,165,21,208
    ,37 166,20,104,133,21,104,13
    3,20 104 1365
41  1530 DATA 133,98,133,100,104
    ,133,97,133 99,230,97,208 2,
    230,98,160,2055
18  1540 DATA 0,130,145,99,32 15
    7,12,240,6,76 132,12,76,72,1
    70,96,1471
37  1550 DATA 32,91,6 165,20,201
    ,8,144,3,76 72 170,141 227,1
    3 32 1405
37  1560 DATA 252,13,32,91,6,32,
    240,13,208,10,173 21,208 45,
    226,13,1503
88  1570 DATA 141,21,208 56 32,9
    1,6,32,101 6,174,227 13 157,
    240 199,1752
07  1580 DATA 32,91 6 32,228 13,
    174 227 13 157,38,208,32 91,
    6,174,1523
FA  1590 DATA 227,13 138 10 170,
    141 227 13,165,20,157 0,208,
    165 21,240,1515
52  1600 DATA 12,173,16 208 13,2
    25,13,141,16,208 76,98 13,17
    3,16 200 1605
FD  1610 DATA 45,226,13 141,16,2
    08,32,91,6,32 101,6,174,227,
    13,157 1488
24  1620 DATA 1,208,32 91 6 32,2
    40,13 208,12,173,25,208,45,2
    26,13 1537
0B  1630 DATA 141,25,208,76,138
    13,173,25,208 13 225 13 141
    25,208,32,1677
70  1640 DATA 91,6,32,240,13 208
    ,12,173,23,208 45,228,13,141
    23 208 1662
57  1650 DATA 76,168 13 173,23,2
    08 13,225,13,141,23 208 32,3
    1,6,32 1445
44  1660 DATA 240,13,208,12 173
    28,208,45 226,13 141,28,208,
    76,157,13,1823
77  1670 DATA 173,28 208 13 225,
    13,141,28 208,32,91,6,32,228
    13,141 1590
F1  1680 DATA 37,208 32 91 6,32,
    228 13,141,38,208,173,21,208
    ,13,225,1674
1C  1690 DATA 13,141 21,208 56 0
    ,0,0,165 20,201,16,144,3,76
    72 1176
FA  1700 DATA 178,165 20,56 165
    20 201 2 144,3,76,72,178,165
    ,20 56,1601
9F  1710 DATA 168,200,56,165 0 4
    2 136,208,252,141,225 13 73,
    255 141,226,2305
28  1720 DATA 13 96 32 253,174,3
    2 212 225 32,91,6,165,20,72,
    165 21 1605
11  1730 DATA 72,32,91 6 166,20,
    164,21,104,133 21,104 133 20
    ,168,20 1276
A2  1740 DATA 32 216,255,88,32,2
    53,174,32,212 225,32,91,6,16
    8 0,166 1991
28  1750 DATA 20,164,21,32,213,2
    55,96,225,120,169,49,141,20,
    3,169,234 1931
6F  1760 DATA 141,21,3,173,28 20
    8,41,254,141,28,208,173,14 2
    20,9 1 1655
CC  1770 DATA 141,14,220 88,96,1
    65,20,201,200,144,3,76,72,17
    8,165,20 1803
4F  1780 DATA 56 165,21,240 9,16
    5,20 201,64,144 3 76 72,178,
    96,32 1582
```

```
1C   17RD DATA 51,R,32,228,13,10,
     10,10,10,133,254,32,51,6,32
     228 11GR
D1   1000 DATA 13,24 10I 254,133,
     254,96,173 17,208,8,32,141,1
     7,208,1RR 1845
CR   1010 DATA 2R,141 24,208,RR,1
     73,17,208,41 223,141,17,200,
     1RR,10,141,165I
RF   1020 DATA 24,208 RR,32,91,6,
     1RS,21,208,2S,165,20,201,25,
     17R,1R 1466
CO   1030 DATA 72,32,91,6,104,170
     ,1RR,21,206,9,1R4,20,152,40,
     176,3 1473
S4   1040 DATA 76,240,255,7R,72,1
     76,0,1RR,0,133,250,169,144,1
     33,251,165,231S
```

```
25   1050 DATA 252,133,174,133,1R
     3,1RR,3,133,175,133,194,165,
     21S 133 252,169,2630
A1   16RD DATA 154,133,253,160,0
     177,250,145,174,230,250,200,
```

```
     2,230,251,230,2647
EC   1070 DATA 174,208,2,230,175,
     165,250 1R7,252,208,234 165,
     251,1R7,253,206,3165
A5   1080 DATA 226,165,114,133,18
     7,169 1SS,133,1R8,165,6,133,
     163,1RR,0,133,2265
06   1090 DATA 185,1GR,0 32,144,2
     $$,1RR,8,76 6I,155,200,608,2
     45,32,207,2146
59   1900 DATA 255,240,2R1,201,49
      240,4,201,RG 48,230,41,15,1
     33,106,76,222R
14   1910 DATA 234 245,147,17,17,
     73 76,60,65,64,32,88,8R,86,7
     3 67,1455
47   1920 DATA RR,32,70 65,77,66
     65,82,13,17,67,65,83,R1,4R,3
     2,945
FA   1930 DATA 47,32 R8,73,83,75,
     61,32,56,3E,76,82,32,57,58,4
     5,R16
E1   1940 DATA 36,0,77,67,70,73,7
     0,68,0,0,0,0,0,0,0,0,464
SD   1950 DATA 0,0,255 255,255,65
     5,255,0,0,0,0,0,0,0,0 0,127R
```

```
PROGRAM  DEMO
```

```
4A   10 REM ** EXAMPLE1 **
CR   20 PRINT"CCLR,DOWN) MOVE MEM
     ORY FROM CHARACTER SET"
SO   30 PRINT" STARTING AT S1200
     (RC800) TO THE"
SG   40 PRINT" BITMAP RCBEEN AT S
     7344 ($E000)"
E2   50 PRINT" IN BLOCKS OF 8  1E
      (1 CHARACTER)"
2D   60 PRINT" THEN FILL REMAININ
     G SCREEN WIIH"
RD   70 PRINT" CHARACTER NO  127
     "
AF   80 GOSUM2R50 REM GET ANY KEY
      TO CONTINUE
AA   90 SYS1044 2,10 REM SET COLO
     URS RED ON L/RED
1C   100 SYS1038 SYS1041 REM CLEA
     R AND COLOUR BITMAP
```

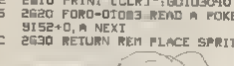
```
03   110 SYS1032 REM BITMAP ON
4D   120 READ ROW COL,RR
6R   130 IFROW=2S6TKENZ30
09   140 FOR C=ITOLEN(RR)
0D   150 SBM=57344+(ROW*320)+(COL
     *8):REM POSITION ON BITMAP
3C   160 CHR=MIDR(RS C,1) CHAR=AS
     C(CHR)
S0   170 IFCHAR>=64TKENCHAR=CHAR-
     R4:REM CONVERT ARC11
C1   160 IFCHAR>=R6TKENCHAR=CHAR-
     64 REM TO CBM DIRPLAY
D8   190 CN=S1200+(CHAR*8) REM WH
     ICH CHARACTER TO PRINT
4A   600 SYS1068,RBM,CN,8:REM MOV
     E MEMORY FROM CHARACTER SET
     TO BITMAP
G7   610 COL=COL+1 NEXT
E4   220 GOTOI20
DD   230 RYRIOSR,2S,100,I,127 SYR
     10SR,300,100,1,127 REM FILL
     BITMAP
8G   240 GETA$ IFA$=""THEN240
S8   250 SYS1035 REM BITMAP OFF
CD   260 PRINT"[CL6]" GOTO430
32   270 REM DATA IN FORM OF (ROW
      COLUMN "TEXT")
CF   280 DATA 6,R,"CSU,S*20,S1)"
3E   290 DATA 7,R,"CS-,SPC20,S-)"
66   300 DATA 8,S,"CS-) THIS IS A
     N EXAMPLE (S-)"
3B   310 DATA S,R,"CS-,SPC20,S-)"
S0   320 DATA 10,R,"CS-)  OF WRIT
     ING TEXTCSPC3,S-)"
07   330 DATA 11,S,"CS-,SPC20,S-)
     "
RD   340 DATA 12,S,"CR-) TO A BIT
     MAP SCREEN (S-)"
31   350 DATA 13,S,"CS-,SPC20,S-)
     "
66   360 DATA 14,S,"CS-)  USING T
     HE MEMORY  (S-)"
D8   370 DATA 15,S,"CS-,SPC20,S-)
     "
0R   380 DATA 1R,R,"CS-,SPC4)MOVE
      COMMAND CSPC3,S-)"
CS   390 DATA 17,S "CS-,SPC20,S-)
     "
3D   400 DATA 18,S,"CSJ,S*20,SK)"
71   410 DATA 256,256,"" REM END
     OF DATA
20   420 REM ** EXAMPLE2 **
DA   430 PRINT"CDOWN) USING THE A
     BOVE RITMAP"
8E   440 PRINT" HERE IS AN EXAMPL
     E OF USING"
66   450 PRINT" THE SPLIT-SCGEEN
     COMMAND TO"
43   460 PRINT" ROLL THE BITMAP U
     P AND DOWN"
OF   470 PRINT" THE SCREEN;"
8D   480 PRINT" PRERR ANY KEY TO
      END DEMO "
EA   490 GOSUB2990
S9   500 SYS1023 REM RASTEG INTER
     BUPT ON
97   510 RYS1041 REM COLOUG BITMA
     P
SC   520 SYS1029,1 REM SET BITMAP
      AT TOP OF SCREEN
C9   530 GETA$ IFA$=""THEN570
32   540 SYS1029,0 REM SPLIT SCRE
     EN OFF
BC   550 SYS1026 REM RASTER OFF
BA   560 PRINT"CCLR)" GOTOORO
AR   570 FOR R=1 TO 200
I8   580 SYSIO2R,R:REM SCREEN SPL
     IT
D2   590 NEXT
08   600 FORR=200 TO 1STEP-1
F2   610 SYSIO2R,R
EC   620 NEXT
45   R30 GOTO530
64   640 REM ** EXAMPLE3 **
08   R50 PRINT CDOWN) USE MEMORY
     FILL COMMAND TO COLOUR"
C9   RRO PRINI  PARTS OF BITMAP A
     ND THEN USE IT"
AC   R70 PRINT" TO CLEAR PART OF
     THE SCREEN,"
SA   R80 PRINT" PRESSING ANY KEY
     TO GET NEXT PART"
37   R90 PRINI" OF DEMO "
A5   700 GOSUB2S90
C4   710 SYS1041 REM COLOUR BITMA
     P
27   720 SYS1032 REM BITMAP ON
2R   730 GETAR IFAS=""THEN730
EA   740 SYS1044 0,S:REM SET COLO
     URS
D7   750 C=PEEK(2S4) REM 254 IR T
     HE CURRENT COLOUR
26   760 RCREEN=1024+4R152+(5*40)
      REM SCREEN ADDRERR IN BANK
     3 + S LINES DOWN
7S   770 NO=15*40 REM NO=NUMGER O
     F LINES * 40 CHARACTERS TO T
     HE LINE
```

```
F0   780 SYSIO7I,RCREEN,NO,C
AA   790 GETA$ IFAR=""THEN780
C9   000 BITMAP=57344+(320*S) NO=
     320*15 REM CLEAR 15 LINER S
     LINES DOWN
0E   810 SYSIO7I,BITMAP,NO,0 REM
     FILL MEMORY WITH (0)
S4   820 GETA$ IFAR=""THEN820
D1   G30 SYR1035 REM BITMAP OFF
G8   840 PRINT"[CLR]"
DA   G50 REM ** EXAMPLE4 **
A5   G60 PRINT"[DOWN) NOW FOG 90M
     E LINE DRAWING DEMO8"
```

```
4F  870 GOSUB2850:POKE53280,0
C1  880 SYS1044,7,0:SYS1041:SYS1
    030:REM SET COLOUR, COLOUR A
    ND CLEAR BITMAP
72  890 SYS1032:REM BITMAP ON
AF  900 FOR X=0TO319STEP5
07  910 SYS1050,X,0,X,199,1 REM
    DRAW LINE COMMAND
18  920 NEXT
7C  930 FORY=0TO199STEP5
2F  940 SYS1050,0,Y,319,Y,1:REM
    I=PLOT
3B  950 NEXT
BB  960 FORO=0TO199STEP4
01  970 B=319-O
00  980 SYS1050,0,0,B,199,1
B3  990 NEXT
23  1000 FORO=0TO199STEP3
17  1010 B=199-O
A7  1020 SYS1050,0,9,319,0,1
BB  1030 NEXT
5E  1040 FORO=0TO199STEP3
AB  1050 B=199-O
7F  1060 SYS1050,0,0,319,B,0:REM
    O=UNPLOT
B0  1070 NEXT
61  1080 FORO=0TO319STEP4
7E  1090 B=319-O
B5  1100 SYS1050,B,0,O,199,0
D8  1110 NEXT
40  1120 GETA$:IFA$=""THEN1120
E9  1130 SYS1035:PRINT"[CLR]"
40  1140 REM ** EXAMPLE5 **
BF  1150 PRINT"[DOWN]  NOW FOR SO
    ME CIRCLES"
73  1160 GOSUB2890
20  1170 SYS1044,4,3:SYS1039:SYS
    1041:SYS1032
E2  1180 FOR R=1TO129
C2  1190 SYS1053,150,100,R,1:REM
    CILCLE COMMAND (I=PLOT)
3B  1200 NEXT
30  1210 FOR R=129TO1STEP-2
BB  1220 SYS1053,150,100,R,0:REM
    O=UNPLOT
50  1230 NEXT
14  1240 GETA$:IFA$=""THEN1240
35  1250 POKE53280,0
DD  1260 SYS1044,14,7:SYS1041
F3  1270 SYS1039:REM CLEAR BITMA
    P
A0  1280 FORD=1TO100
21  1290 SYS1053,0,0,0,1
5B  1600 NEXT
9E  1310 FORO=1TO100
BB  1320 A=199-O
AC  1330 SYS1053,0,A,0,1
C3  1340 NEXT
7A  1350 FORD=1TO100
C8  1360 X=319-O:Y=199-O
18  1370 SYS1053,X,Y,0,1
EB  1380 NEXT
A2  1390 FORO=1TO100
53  1400 X=319-O
B0  1410 SYS1053,X,0,0,1
I3  1420 NEXT
F2  1430 GETA$:IFA$= "THEN1460
BE  1440 FORD=100TO1STEP-2
D5  1450 X=319-O
4E  1460 SYS1053,X,O,D,0
41  1470 NEXT
5E  1480 FORD=100TO1STEP-2
96  1490 X=319-O:Y=199-O
59  1500 SYS1053,X,Y,0,0
BB  1510 NEXT
19  1520 FORD=100TO1STEP-2
AA  1530 A=199-O
3B  1540 SYS1053,0,A,0,0
95  1550 NEXT
BB  1560 FORD=100TO1STEP-2
00  1570 SYS1053,0,O,D,0
B0  1580 NEXT
20  1590 GETA$:IFA$=""THEN1590
E1  1900 SYS1035:PRINT"[CLR]"
BB  1610 SYS1035:PRINT"[CLR]"
A1  1620 REM ** EXAMPLE6 **
0E  1630 PRINT"[DOWN]  DRAW SOME
    SHAPES ON SCREEN AND"
AE  1640 PRINT" USE THE FILL AND
    PATTERN FILL COMMANDS,"
EA  1650 PRINT" FINALLY ERASING
    EVERYTHING AGAIN"
B1  1660 GOSUB2890
```

```
B2  1670 SYS1044,2,3:SYS1041:SYS
    1039
0E  1680 SYS1032
FF  1690 M=1:REM 1=PLOT 0=UNPLOT
10  1700 SYS1050,10,35,35,10,M:R
    EM DRAW LINE COMMANDS
BE  1710 SYS1050,35,10,60,35,M
B3  1720 SYS1050,50,35,35,60,M
62  1730 SYS1050,35,60,10,35,M
F5  1740 SYS1050,250,10,310,10,M
5E  1750 SYS1050,310,10,310,60,M
BB  1760 SYS1050,310,60,250,60,M
3A  1770 SYS1050,250,60,250,10,M
58  1780 SYS1050,260,165,285,140
    ,M
B0  1780 SYS1050,285,140,310,165
    ,M
43  1900 SYS1050,310,185,285,180
    ,M
3F  1910 SYS1050,285,180,260,165
    ,M
21  1920 SYS1050,10,140,80,140,M
4D  1930 SYS1050,60,140,60,180,M
57  1940 SYS1050,60,160,10,180,M
19  1950 SYS1050,10,180,10,140,M
A1  1960 SYS1053,110,100,40,M:RE
    M DRAW CIRCLE COMMANDS
75  1970 SYS1053,160,60,40,M
EF  1980 SYS1053,210,100,40,M
F2  1990 SYS1053,160,150,40,M
0C  1900 SYS1053,160,100,60,M
AD  1910 SYS1058,110,100,M:REM F
    ILL AREAS OF BITMAP
0C  1920 SYS1056,160,50,M
FA  1930 SYS1056,210,100,M
D1  1940 SYS1056,160,150,M
24  1950 SYS1059,35,35,M,49:REM
    FILL WITH PATTERN
7E  1960 SYS1059,285,35,M,60:REM
    FILL NO 5 REPRESENT
33  1970 SYS1058,285,165,M,51:RE
    M (I,2,3,4) RESPECTIVELY
AE  1980 SYS1059,35,155,M,52:REM
    SEE SCREEN DISPLAY CODES IN
    USER GUIDE
0C  1990 SYS1058,85,100,M,102
23  2000 SYS1058,255,100,M,102
FF  2010 SYS1058,5,35,M,102
E1  2020 SYS1058,315,35,M,102
45  2030 SYS1058,319,195,M,102
BB  2040 SYS1058,5,185,M,102
4F  2050 SYS1058,180,100,M,102
8E  2060 IFM=1THENSYS1068,40880,
    57344:@000:REM STORE BITMAP
    UNDER BASIC ROM
I0  2070 GETA$:IFA$=""THEN2070
5E  2080 IFM=1THENM=0:GOTO1700:R
    EM SET MODE TO 0=UNPLOT
B0  2090 SYS1035:PRINT"[CLR]"
B2  2100 REM ** EXAMPLE7 **
A4  2110 PRINT"[DOWN] MOVE THE P
    REVIOUSLY STORED BITMAP'
DA  2120 PRINT" ( SEE LINE 2130
    IN ABOVE DEMO)"
4C  2160 PRINT" FROM THE STORAGE
    SPACE UNDER THE BASIC"
EE  2140 PRINT" ROM BACK INTO TH
    E BITMAP SCREEN'
55  2150 GOSUB2890
BC  2160 SYS1041:SYS1032
45  2170 SYS1068,57344,40960,800
    0:REM MOVE DATA
BA  2180 GETA$:IFA$=""THEN2180
51  2190 SYS1035:PRINT"[CLR]"
5D  2200 REM ** EXAMPLE8 **
E7  2210 PRINT"[DOWN] MOVE BITMA
    P DATA TO MEMORY AT'
B0  2220 PRINT" 24576 ($6000) AN
    D THEN SAVE IT TO DISK"
51  2230 PRINT" THEN CLEAR BITMA
    P AND LOAD THE SAVED"
53  2240 PRINT" DATA DIRECTLY IN
    TO THE BITMAP SCREEN"
F1  2250 PRINT" AT 57344 ($E000)
    "
D2  2260 PRINT" PLEASE ENSURE TH
    AT THE DRIVE IS READY"
53  2270 PRINT" BEFORE CONTINUIN
    G WITH THIS DEMO '
28  2280 PRINT" IF YOU WANT TO S
    KIP THIS DEMO PRESS <S>"
FB  2290 GOSUB2890
8E  2300 IF A$="S"THEN2350
55  2310 SYS1032:SYS1041
C8  2320 SYS1068,24576,57344,800
    0:REM MOVE BITMAP DATA TO ME
    MORY STARTING AT 24576
BB  2330 SYS1039:REM CLEAR BITMA
    P
91  2340 SYS1074,"DATA",08,00,24
    575,32576:REM SAVE MEM  FROM
     24576 TO 24576+8000
2F  2350 SYS1077 "DATA",08,00,57
    344:REM LOAD DATA DIRECTLY I
    NTO BITMAP
E0  2350 GETA$:IFA$="" THEN2390
C4  2370 SYS1035:PRINT"[CLR]"
17  2390 REM ** EXAMPLE9 **
8E  2390 PRINT"[CLR,DOWN] NOW FO
    R A DEMONSTRATION OF THE SOU
    ND"
25  2400 PRINT" AND SPRITE SETUP
    COMMANDS"
C0  2410 GOSUB2890:POKE53280,0:P
```

```
    OKES3291,0
65  2420 SYS1080,19,0 PRINT"{RVS
    ON}{SPC8}{RVSOFF,SPC10,RVSON}
    {RVSOFF}"
5B  2430 SYS1080,20,28 PRINT"{CC
    ,SPC8 SE}" REM PRINT AT ROW=
    20, COLUMN=28
0A  2440 SYS1080,21,28 PRINT" {R
    VSON,CI,RVSOFF,CI6,RVSON,CI,
    RVSOFF} "
BE  2450 SYS1080,23,8 PRINT"{SPC
    28}"
09  2460 SYS1023 FIN=0 X=40 Y=25
    :GOSUB2820
E9  2470 V1=PEEK(1128) REM TEST
    VOICE1
03  2480 IF V1=0 THENGOSUB2720 R
    EM IF OFF THEN READ NEXT NOT
    E DATA
47  2490 IF MC=2 THEN5=0 REM USE
     DELAY TO CONTROL SPRITE MOV
    EMENT
63  2500 X=X+(S/50) Y=Y+(S/75) R
    EM LONGER NOTE PLAYS FASTER
    SPRITE MOVES
12  2510 SYS1065,0,1,0,7,X,Y,1,1
    ,1,1,MC REM SPRITE SETUP COM
    MAND
D0  2520 IFFIN<>1THEN2470 REM IF
     MORE NOTES
20  2530 FORD=0TO10
B0  2540 Y=Y+1
1G  2550 SYS1065,0,1,0 7 X,Y,1 1
    ,1,1 0 REMSPRITE SETUP COMMA
    ND
B9  2560 NEXT
DC  2570 V3=PEEK(1130) IF V3<>0T
    HEN2570 REM WAIT TILL VOICE
    3 FINISHED PLAYING
DF  2580 SYS1028 REM TURN OFF IN
    TERRUPT
17  2590 GOSUB2990
AE  2600 SYS1065,0,0
92  2610 PRINT"{CLR}":GOTO3040
F5  2620 FORD=0TO63 READ A POKE4
    9152+D,A NEXT
BC  2630 RETURN REM PLACE SPRITE
```



```
    DATA IN SPRITE 0 (45152-)
20  2640 DATA 000,000,000,000,00
    0,000,000,000
C3  2650 DATA 000,000,000,000,00
    0,080,000,002
40  2660 DATA 150,128,042,170,16
    8,170,170,170
78  2670 DATA 170,170,170,238,23
    0,230,187,187
99  2680 DATA 187,170,170,170,04
    2,170,168,009
01  2690 DATA 170,144,005,000,09
    0,020,000,020
57  2700 DATA 019,000,004,064,00
    0,021,000,000
F7  2710 DATA 000 000,000,000,00
    0,000,000,000
D7  2720 READ LF,HF,D,MC REM GET
     NOTE DATA PLUS MULTICOL 2(S
    EE LINE 3542)
6C  2730 IFLF=-259THEN2,80 REM TE
    ST FOR END OF DATA
5E  2740 SYS1062,1,1S,LF,HF,0,24
    0,0,3,6S,0 REM VOICE 1
FC  2750 SYS1062,2,15,43,137,128
    ,255,0,0 33 100 REM VOICE2
0E  2760 SYS1062,3,15,149,00,128
    ,255,0,0,125,100 REM VOICE3
D9  2770 GOTO2790
70  2780 FIN=1 REM @IGNAL NO MOR
    E DATA SEE LINE 3550
7E  2790 RETURN
E1  2800 DATA 53 7,75,2 REM NOTE
     DATA
4F  2810 DATA 23,0 75,1
B7  2820 DATA 100 6,75,2
C9  2830 DATA 0,0,15,1
90  2840 DATA 54,3,75,2
2C  2850 DATA 200,4,100,1
46  2860 DATA 53,7,50,2
ED  2870 DATA 23,0,50,1
30  2880 DATA 100 6,50,2
94  2890 DATA 0,0,10,1
AB  2900 DATA 54,3,50,2
05  2910 DATA 200,4,75,1
7B  2920 DATA 107,14,20,2
0B  2930 DATA 47,16,20 1
72  2940 DATA 216,12,20,2
9A  2950 DATA 0,0 5,1
BB  2960 DATA 109,6 20,2
9D  2970 DATA 155,3,40,1
F9  2980 DATA 256,0,0,0
63  2990 SYS1080,23,0 REM SET CU
    RSOR AT 23 LINE, 0 CHARACTER
    S ALONG
9A  3000 PRINT"{RVSON}PRESS ANY
    KEY TO CONTINUE{RVSOFF}"
0C  3010 GET@$ IFA$=""THEN3010
59  3020 RETURN
9A  3030 REM ** MEM/MAP AND COMM
    ANDS PRINTOUT **
BC  3040 PRINT"{CDOWN} THIS PART
    OF THE PROGRAM PRINTS OUT"
76  3050 PRINT" A MEMORY MAP AND
     A LIST OF THE NEW"
E1  3060 PRINT" COMMANDS  SO IF
    YOU HAVE A PRINTER"
80  3070 PRINT" MAKE SURE IT IS
    READY NOW."
2C  3080 PRINT" IF YOU DON'T HAV
    E A PRINTER THEN YOU"
C2  3090 PRINT" WILL JUST HAVE T
    O REFER TO THE MAGAZINE"
I7  3100 GOSUB2990
AC  3110 OPEN2,4 S1$=CHR$(15):@S
    $=CHR$(8)
```

```
EA  3120 PRINT#2,CHR$(14)"{SPC13
    }NEW MEMORY MAP"
AB  3130 PRINT#2,CHRR(14)"{SPC12
    ,CY18}"S1@
FA  3140 PRINT#2
D5  3150 PRINT#2,S1R"  {CA,S*18,
    C9}"@S$
04  3160 PRINT#2,S1$"  {CQ,S*18,
    CW}-192 BYTES STORAGE"B$
F7  3170 PRINT#2,S1R"  {S-,SPC18
    ,S-}"@S$
3S  3180 PRINT#2,S1$"  {S-,SPC18
    ,S-}"@S$
CE  3190 PRINT#2,S1R"  {S-,SPC18
    ,S-}-@ITMAP SCREEN (57344 -
    65344)"RS$
A5  3200 PRINT#2,S1$"  {S-,SPC18
    ,S-}"@S$
08  3210 PRINT#2,S1@"  {S ,SPC18
    S-} OR STORAGE AREA"@S$
0I  3220 PRINT#2,S1$'  {S-,SPC18
    ,S-}"@S$
```

```
D8  3230 PRINT#2,S1@"  {CQ,R*18,
    CW}"@S$
C0  3240 PRINT#2,S1@'  {S-,8N18,
    S-}"@S$
F9  3250 PRINT#2,S1$"  {S-,5N18,
    S-}-UNUSABLE ( VIC, SID, I/O
    ETC  )"@S$
81  3260 PRINT#2,S1@"  {S-,5N18,
    S-}"@S$
73  3270 PRINT#2,S1@"  {CQ,6*18,
    CW}"@S$
4A  3280 PRINT#2,S1$   {S-,SPC18
    S-}-CHARACTER SET (51200 -
    53248){2SS "
0E  3290 PRINT#2 "CHARS  * 8 BYT
    ES)"B$$
31  3300 PRINT#2,S1$   {S-,SPC18
    ,S-}"@S$
BC  3310 PRINT#2,S1$"  {CQ,S*18
    CW}-SPRITE POINTERS (51192 -
    51199)"@S$
71  3320 PRINT#2 S1$"  {CQ S*18,
    CW}"BS$
72  3330 PRINT#2,S1$"  {S-,SPC18
    ,S-}-SCREEN MEMORY (50176 -
    51175)"B$$
B6  3340 PRINT#2,S1$"  {CQ,S*18,
    CW}"@S$
1A  3350 PRINT#2,S1$"  {S-,SPC18
    ,S-}-DISPLAY SPRITE STORAGE
    (0-15) (45152"
AC  3360 PRINT#2," - 50175)"R$@
F8  3370 PRINT#2,S1$"  {CQ,S*18,
    CW}"@S$
DF  3380 PRINT#2,S1$"  {S-,SPC18
    ,S-}-SPRITE STORAGE (128 SPR
    ITES)"B$$
9C  3390 PRINT#2,S1$"  {S-,6PC18
    ,S-}"@S$
82  3400 PRINT#2,S1$   {S-,SPC1R
    ,S-} OR CHARACTER SETS (4)"8
    $@
20  3410 PRINT#2,S1R'  {S-,SPC18
    ,S-}"@S$
DF  3420 PRINT#2,S1R"  {S-,SPC18
    ,S-} OR @ITMAP STORAGE (1)"8
    $@
04  3430 PRINTR@,@1$'  {S-,SPC18
    ,S-}"@S$
60  3440 PRINT#2,S1$'  {S-,SPC18
    ,S-} OR OTHER (40950 - 45151
    )"B$R
60  3450 PRINT#2,S1@"  {CQ,S*18,
    CW}"@S@
17  3460 PRINT#2,S1@"  {S-,SPC18
    ,S-}-TOP OF BASIC (40959)"@S
    $
EC  3470 PRINT#2 S1$"  {S-,SPC10
    ,S-}"@S@
BE  3480 PRINT#2 S1$"  {S-,SPC18
    S-}"@B$
3@  3490 PRINT#2,S1R"  {S-,SPC18
    ,S-}"B$R
```

```
A8  3500 PRINT#2,S1$"  {S-,$N,$N
    ,$N,$N,$N $N $N,$N,$N,$N,$N,$N,
    $N $N,$N,$N,$N,$N,$N S-}"
5A  3510 PRINT#2,S1$"{SPC3}MAIN
    @BASIC MEMORY    37182 BASIC
    BYTES FREE"
53  3520 PRINT#2,S1$"  {S-,$N,$N
    ,$N,$N,$N,$N $N $N,$N,$N,$N,$N,
    $N,$N,$N,$N,$N,$N,$N,S-}"B$$

80  3530 PRINT#2,S1$"  {S-,$PC18
    ,S-}"B$$
C2  3540 PRINT#2,S1$"  {S-,$PC18
    ,S-}"B$$
FC  3550 PRINT#2,S1$"  {S-,$PC18
    ,S-}"B$$
79  3560 PRINT#2,S1$"  {S- $PC18
    ,S-}-START OF BASIC (3755)"B
    $$
50  3570 PRINT#2,S1$"  {CQ,S*18
    CW}"S$$
9D  3580 PRINT#2,S1$"  {CO,S*18,
    CW}-NEW COMMANDS PRODRAM (10
    20 -3754) "S$$
A0  3590 PRINT#2,S1$"  {S-,$N18,
    S-}"S$$
1C  3600 PRINT#2 S1$"  {CZ,S*19,
    CX}- OPERATING SYSTEMS AREA"
    B$$
```

```
9C  3610 PRINT#2,S1$
5E  3620 PRINT#2,"  CURRENT COLO
    UR STORED IN - (254)" PRINT#
    2
93  3630 PRINT#2,"  VOICE CONTRO
    LS (PEEK TO FIND OUT IF VOIC
    E STILL ON)"
B9  3640 PRINT#2,"I E  0=OFF  IO
    0=ON"
45  3650 PRINT#2  {SPC5}V1-1128,
    V2-1129, V3-1130"
B5  3660 PRINT#2 PRINT#2,"  PIXE
    L ON OR OFF - PEEK(02) I.E-
    0=OFF, 100=ON"
B8  3670 GOSUB2380
5E  3680 PRINT#2,CHR$(14)"{SPC12
    }LIST OF COMMANDS'
FF  3690 PRINT#2,CHR$(14)"{SPC11
    ,CYIS}"S1$
4C  3700 PRINT#2
56  3710 PRINT#2,"RECONFIGURE -
    SYS1020"
B5  3720 PRINT#2,"{CT11}"
57  3730 PRINT#2,"RASTER INTERRU
    PT ON - SYS1023"
II  3740 PRINT#2,"{CT18}"
08  3750 PRINT#2,"RASTER INTERRU
    PT OFF - SYS1020
9C  3760 PRINT#2,"{CT20}"
CA  3770 PRINT#2,"SPLIT-SCREEN -
     SYS1029,0 OR 1-200"
3B  3780 PRINT#2,"{CT12,SPC10}{O
    =OFF, 1-200=SCREEN LINE)"
78  3790 PRINT#2
FC  3800 PRINT#2  BITMAP ON - SY
    S1032"
E0  3810 PRINT#2,"{CT5}"
C8  3820 PRINT#2,"BITMAP OFF - S
    YS1035"
78  3830 PRINT#2,"{CT10}"
55  3840 PRINT#2,"CLEAR BITMAP -
     SYS1030"
78  3850 PRINT#2,"{CT12}"
AE  3860 PRINT#2,"COLOUR BITMAP
    - SYS1041"
09  3870 PRINT#2,"{CT13}"
B5  3880 PRINT#2,"SET CURRENT CO
    LOUR - SYS1044,0-15,0-15"
80  3890 PRINT#2,"{CT18 $PC10}{O
    RAW}{BACKGROUND}"
05  3900 PRINT#2
59  3910 PRINT#2,"PLOT POINT ON
    BITMAP - SYS1047,0-319,0-199
    ,0-1-2"
F7  3920 PRINT#2,"{CT20,SPC10}{X
    -COORD}{Y-COORD}{MODE 0=UNLO
    T,"
```

```
C5  3930 PRINT#2,"1=PLOT,2=PEEK)
    "
1D  3940 PRINT#2
03  3950 PRINT#2,"DRAW LINE - SY
    S1050,0-319,0-199,0-319 0-19
    9,0-1"
F0  3960 PRINT#2 "{CT3,SPC10}{X1
    }{Y1}{X2}{Y2}{MODE}"
3B  3970 PRINT#2
0C  3980 PRINT#2,"DRAW CIRCLE -
    SYS1053,0-319,0-199,1-129,0-
    1"
ED  3990 PRINT#2,"{CT11,SPC10}{C
    X-COORD}{CY-COORD}{RADIUS}{M
    ODE}"
69  4000 PRINT#2
4F  4010 PRINT#2,"FILL BITMAP -
    SYS1056,0-319,0-199,0-1"
61  4020 PRINT#2,"{CT11,SPC10}{X
    -COORD}{Y-COORD}{MODE}"
R7  4030 PRINT#2
EB  4040 PRINT#2,"FILL WITH CHAR
    ACTER - SYS1059,0-319,0-199,
    0-1,0-255"
```

```
51  4050 PRINT#2 "{CT15,SPC10}{X
    -COORD}{Y-COORD}{MODE}{CHAR
    ND}"
A5  4060 PRINT#2
57  4070 PRINT#2,"SET SOUND - SY
    S1062,1-3,0-15,0-255,0-255,0
    -255,0-255,0-255 ",
A7  4080 PRINT#2,"0,15,0-255,0-2
    55"
```

```
3C  4090 PRINT#2 "{CT19,SPC8}{VOI
    CE}{VOLUME}{LOW FREQ}{HIGH F
    REQ}{A/D}{S/R}"
EI  4100 PRINT#2 "{SPC10}{LOW PU
    LSE}{HIGH PULSE}{WAVEFORM}{D
    URATION}'
B8  4110 PRINT#2
59  4120 PRINT#2,"SET SPRITE - S
    YS1065,0-7,0-1,0-255,0-15,0-
    320,0-255,0-1,0-1";
B0  4130 PRINT#2,",0-1,0-15,0-15
    "
94  4140 PRINT#2,'{CT10,SPC10}{S
    PRITE NO}{OFF/ON}{SPRITE POI
    NTER}{COLOUR}",
23  4150 PRINT#2,"{X POS}{Y POS}
    "
83  4160 PRINT#2,"{SPC20}{X EXP}
    {Y EXP}{MULTI COL OFF/ON}{MC
    1}{MC 2}"
F4  4170 PRINT#2
E5  4180 PRINT#2,"MOVE MEMORY -
    SYS1068,0-65535,0 65535_0-32
    767"
51  4190 PRINT#2,"{CT11 $PC10}{A
    DDRESS MOVE TO}{ADDRESS MOVE
    FROM}",
2F  4200 PRINT#2 "{LENGTH}"
4C  4210 PRINT#2
61  4220 PRINT#2 "FILL MEMORY -
    SYS1071,0 65535_0-32767,0-25
    5"
E7  4230 PRINT#2,"{CT11,SPC10}{S
    TART ADDRESS}{LENGTH}{NUMBER
    }"
6A  4240 PRINT#2
28  4250 PRINT#2,"SAVE MEMORY -
    SYS1074, {SPC4}' 08,00,0-655
    39,0-65535"
94  4260 PRINT#2,"{CT11 $PC10}{F
    ILE NAME}{DEVICE}{OO}{MEMORY
     START}",
29  4270 PRINT#2,"{MEMORY FINISH
    }"
82  4280 PRINT#2
10  4290 PRINT#2,'LOAD MEMORY -
    SYS1077,'{SPC4} ,08,00,0-655
    35"
CB  4300 PRINT#2,'{CT11,SPC10}{F
    ILE NAME}{DEVICE}{OO}{LOAD A
    DDRESS}'
B0  4310 PRINT#2
3D  4320 PRINT#2 "SET CURSOR POS
    - SYS1080,0-24,0-39"
EE  4330 PRINT#2,"{CT14,SPC10}{R
    OW}{COLUMN}"
50  4340 PRINT#2 CLOSE2 PRINT {C
    LR}".END
```

DiskOS

Accessing the disk drive is child's play through the square window

DiskOS is an operating system which employs windows to ease communications with a disk drive Its ingenious routines interrupt without interrupting! After calling up DiskOS, programs can resume as though nothing had happened

Whenever you need to use one of the functions of DiskOS just press the CBM key with the CTRL key and a menu will appear at the top of the screen This may be done when running Basic or machine-code programs and should be compatible with most of them because it does not use the IRQ interrupts

When selecting options use the first capital letter of the menu name or option After a command is complete, press the spacebar to get back to the opening menu

The QUIT option on all the menus will return you to the start-up menu at the top of the screen

Menu 1 – Info

Press 'I' for Info and the menu will appear Just one item is contained on this one – a short note about the program

Menu 2 – Disk

This allows access to the disk commands

DIR displays the directory which, unlike LOAD"$",8, does not overwrite a Basic program.

ERROR reads the disk status (if the red LED flashes)

INIT reads in the disk information after a disk change

VAL is the validate command which cleans up the disk, and should always be used after SCRATCHing files

FORMAT is the same as the usual Basic command OPEN 1,8,15,"NEW TEST DISK,64" CLOSE 1

RENAME will change a file name to another name. just type NEW NAME=OLD NAME and press RETURN

COPY will copy a file from one disk to another and will ask for the "source" disk which holds the original and the 'destination' disk onto which the copy is made

Menu 3 – Misc

KILL This returns the C64 to normal and disables the use of CTRL+CBM SYS 49152 will restore it

EXIT will let the computer carry on from where you interrupted it by pressing CTRL+CBM

Menu 4 – Screen

COLOURS allows you to change the screen colours and the new setting will be maintained until changed again or SYS 49152 is called to restore the default values

DUMP#4 will dump the text screen at $0400 to the printer If you design a screen using the cursor keys in Basic and go to the screen menu you can then print it out with this facility

```
PROGRAM: DISKOS.BAS

DE  10 REM DISKOS
9D  20 REM TYPE IN IM15 PROGRAM
    AND SAVE IT ON A SPARE TAPE
    OR CASSETTE
DD  30 REM WHEN YOU RUN IT MAKE
    SURE YOU HAVE CUED UP THE TA
    PE/DISK FOR SAVING DISKOS
3D  40 REM WHEN USING DISKOS- LO
    AD USING ,8,1 AND TYPE NEW
6D  50 REM TYPE SYS 49152 TO GET
     THE DISKOS MESSAGE
48  60 BL=252   LN=50    SA=4915
    2
A2  70 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A
65  80 POKE53280,A POKE SA+L*16+
    D,A NEXT D
51  90 READ A IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
4A  100 NEXT L
DF  110 DATA 169,167,160,192 32,
    30,171,169,14,141,32,208 169
    ,6,141,33,1034
74  120 DATA 208 120,188,30 141,
    143,2,169,192 141 144,2 00 9
    6,173,141,1958
7A  130 DATA 2,201,6,240,3,76,72
    ,235,120,169 72,141,143,2,16
    8,235,1006
05  140 DATA 141 144,2,88,173,24
    208,141,63 3 169,23,141,24
    208 169 1721
F1  150 DATA 8 32,210,255,56,32
    240,255,140 60 3,142,61 3,16
    5,0 1666
45  160 DATA 133,255,32,159,192
    32,115 192 169,1,133,255 32,
    199,192,24,2156
D7  170 DATA 172,60,3 174,61 3,3
    2,240,255 32,17,192,173,63,3
    ,141,1621
80  180 DATA 24,208 96,169,1,133
    ,204,169,5,160,153 32,30,171
    32 220,1655
75  190 DATA 255 240,251-201 73,
    208,3 76,107,153 201,68,208,
    3,76 72,2235
52  200 DATA 154 201 77,208,3,76
    ,88,155,201 83,208 3-78-169,
    204,76,2063
E0  210 DATA 126,192,169,0,133,1
    98,56 154 147,17,32,68 73 03
    ,75,78,1642
94  220 DATA 83,32,86,45,46 50,3
    2,40 85 83 69,32,67,84,82,76
    ,996
80  230 DATA 43,67,66,77 41,13 0
    ,169 0 133,251,133,253,188,4
    ,133,1552
C7  240 DATA 252,169,160,133,254
    ,169,255,208 20,160,0,177,25
    1 145,253,200-2802
```

by Gary Mayhew

```
A2  250 DATA 200,249,230,252,230
    ,254,165,252,201,8,208,239 9
    6,180,0,198 2950
11  260 DATA 1,177,253,145,291,2
    00,208 249,230,254,230 252,1
    65,252 201,8 3079
48  270 DATA 208,235,230 1,56,14
    7,147 10,176,192,192,192 192
    ,192,192 178,2552
59  280 DATA 192,192 152,192,192
    ,152,178,192 192,192,192,192
    ,192,178 192 192,3044
04  290 DATA 192 192,192,192,192
    ,192,174,13,10,221,92,201,78
    ,70 79 32 2070
90  300 DATA 221,32,186,73,83,75
    ,32,221,32 205,73,83,97,32,2
    21,32 1978
EE  310 DATA 211,67,92,89,69,78,
    32,221 13 10 173,152,192,192
    ,192 192,1953
45  320 DATA 192,177,192 192 192
    ,192,192,192,177 192 192,192
    ,192,192,192,177 3027
2D  330 DATA 192,192,192 192,192
    ,192,192,192,169,13,0,169,11
    7,160 133,32,2403
81  340 DATA 30,171 79,173,153,1
    9,17,17,10,171,192,192,192,1
    92,192,192 2037
40  350 DATA 215,13,10,221,193,6
    6,79,85 84,32,221,13,10,173,
    192,152,1819
42  360 DATA 192,192,152,192,192,189
    ,13,0,32,228,255 240,251,201
    ,85,240,12 2518
92  370 DATA 201,78,208,243 104,
    104,76 115,192 76,151,153,96
    32,228,255,2352
E1  380 DATA 240,251,201,65,208,
    247,76,187,193 208,220,169,2
    00,160,193,32,2850
3F  390 DATA 30 171,32,176,157,7
    6,115,192,10,17,17,17,17 17,
    17,17,1127
8C  400 DATA 17,17 32,32 10,176,
    192 192,192 192,192 192,192,
    192,152,192,2212
7E  410 DATA 192 192 192 192,192
    ,192,192,192 192 192,192 192
    ,192 192,192 152 3072
A0  420 DATA 192 192 192 192,192
    ,192 192,174,13,32,32,10,221
    ,32 186 73 2135
7E  430 DATA 83 75,207,211,32 86
    ,49,46 50,32 86,83,82,199,49
    ,205,1508
88  440 DATA 83,83,72,83,87 32,2
    02 85 78,32,39,56 56,32,221
    13,1208
33  450 DATA 32 32 10,173,192,19
    2 192 192 192 192,192,19 2,
    192,192,192,2559
8C  460 DATA 192,192 192,192,152
    ,192,192,192,192 192,192,192
    ,192,192,192 192,3072
D4  470 DATA 192 192,192 192,192
    ,109,13,0,169 83 190 194,32,
    30 171 76,2077
21  480 DATA 10,195,96,19,17 17,
    29,29,29,29,29,29,29 10,219,
    192 894
11  490 DATA 192,192,192,192,192
    ,219,13,29,29,29,29,29,29,29
    ,10,221,1634
6D  500 DATA 156,73,52,49,32 32
    221,13,29,29,29,29,29,29,29,
    10,515
69  510 DATA 221,197,82,82,79,82
    ,32 221,13,29,29,29,29,29,29
    29,1212
9C  520 DATA 10,221,201,70,73,04
    ,46 32 221,13,29,29,29,29,29
    ,29,1161
D0  530 DATA 29,10,221 214,65,78
    46,32 32,221,13,29,29,29,29
    29 1112
07  540 DATA 29,29,10,221,211,67
    82,46,32 32 221,13,29,29 29
    ,29 1117
23  550 DATA 29,29,29,10,221,198
    ,79,82,77,65,84,221 13,29 29
    ,29,1232
A8  560 DATA 29,29,29,29,10,221,
    210,65,78,65,77,89,221,13,29
    ,29,1215
50  570 DATA 29,29,29,29,29,10,2
    21,195,73,80,85,32 32,221,13
    ,29,1154
36  580 DATA 29,29,29,29,29,29,1
    0,221,209,85,73,84,32,32,221
    ,13,1162
74  590 DATA 29,29,29 29 29,29,2
    9,10,173 192,192,192 192,192
    ,192 185,1735
7D  600 DATA 13,0,32,228,255,240
    ,251,201,91,209,3,76,115,192
    ,201,86 2164
C1  610 DATA 208,3,76,17,156,201
    ,69 208,3,76,212,156,201,73,
    208,3,1350
DA  620 DATA 76,8,196,201 86 209
    ,3,76,32,196,201,83,208,3,76
    ,16,1573
58  630 DATA 200,201,70,208,3,76
    ,138,200,201,82,209,3,76,220
    ,200,201,2281
F5  640 DATA 67 208,3,76,40,201,
    75,10 195,165,100,160,195,32
    ,30,171,1749
E9  650 DATA 76,224,195,96,19,17
    ,17,29,29 29,29,29,29,29,29,
    29 505
0A  660 DATA 29,29,29,29,29,10,2
    19 192,192,192,192,192,2
    15,13,29 1795
10  670 DATA 29,29,29,29,29,29,2
    9,29,29,29 29,29,29,10,221,2
    03,819
EA  680 DATA 73,76,79,32,32,221,
    13 29,29,29,29,29,29 29,29,2
    9 784
DF  690 DATA 29,29,29,29,29 10,2
    21,197,90,73,84,32,32,221,13
    ,29,1153
14  700 DATA 29,29,29,29,29,29,2
    9,29,29,29,29,29,29,10 221 2
    05,825
81  710 DATA 05,73,84,32,32,221,
    13 29,29,29,29,29,29,29,29,2
    9,901
5C  720 DATA 29,29,29,29,29 10,1
    73 192,192 192,192,192,192,1
    89,13,0,1690
7A  730 DATA 32,228,255,240,251,
    201 81 208 3,76,116,192 201,
    69,208,6,2365
5C  740 DATA 165,9,32,210,255,99
    ,201 79,208 230,165,14,141,3
    2,208,165 2210
E0  750 DATA 6,141,33 208,169 21
    ,141,24,208 169,8 32 210,255
    ,79,148,1850
2E  760 DATA 227,189,140 190,196
    ,32 30,171 169 38,133,251,19
    9 251,133,187 2462
E8  770 DATA 169,0,133 188,189 1
    ,133 183 169,8 133,186 189,9
    6,133,185,2055
54  780 DATA 32,213,243,165,188,
    32 189 255 195 165,32 150,25
    5,165,0,133 2395
AB  790 DATA 144,160,3,132 251,3
    2,195 255 133,252,173,141,2,
    201 1,240 2205
DD  800 DATA 248 164,144,200 47,
    32 165,255,164,144,200,40 16
    4,231,136,200,2575
D6  810 DATA 229,186,252,32,205,
    189,189 32,32,210,255 32 165
    255,166,144,2530
3F  820 DATA 208 10 170 240,6,32
    ,210,255 76 107,196,169 13,3
    2,210 255,2197
A5  830 DATA 160,2 208 151,32,86
    ,246,169,180 160,196 32,30,1
    71,32,176 2051
B7  840 DATA 157,75,115,192 147,
    200,79,75,69,32,211 200,201
    198 212,32,2235
15  850 DATA 84,79,32,208,85,85,
    83,69,32,204,73,83,84,73,78,
    71,1403
81  860 DATA 45,17 13,0 17 208,8
    2,69,83,83,32,39,211 80,85,6
    7 1112
B6  870 DATA 68,39,32,70,79 82,3
    2 156 73 83,75,207,211 32,20
    5,65,1554
F3  880 DATA 78,85 46,0,24,160 0
    ,162,21 32 240,255 169,23,16
    0,197 1652
E6  890 DATA 32,30,171,169,15 15
    2,8 160 15 32 186,255,169 0
    32,189 1625
15  900 DATA 255 32,152 255 152,
    15,32,156,255,32,207,255,201
    ,13,240,5 2349
5A  910 DATA 32,210 255,208,244,
    169,15,32 195 255,32,204,255
    ,32,176,197 2511
4F  920 DATA 76,119,192,32,176 1
    97,96,32,18,176 192,192,192,
    192,192,192,2282
44  930 DATA 192,192,192,192,192
    ,192,192,192,192 192,192,192
    ,192,192,192,192 3072
DO  940 DATA 192 192 192,192,192
    ,192,192 192 192,192,192,192
    ,192,192,174,13,2875
87  950 DATA 92,15 221,211,84,89
    ,84,85,83,58,32 32,32,92,32
    32 1133
94  960 DATA 32,32,92 32,32,32,3
    2,32,32 32,32,32,32,32,32 32
    ,512
CD  970 DATA 32,32,32 32 32,32,3
    2,221 13,32,10,173,192,192,1
    92,192,1449
```

```
B2  880 DATA 192,192,192,192,192,192
    ,192,192,192,192,192,192,192,192,192,192,192,192,192,192,3072
88  890 DATA 192,192,192,192,192,192
    ,192,192,192,192,192,192,192,192,192,192,192,192,192,192,3072
6E  1000 DATA 185,157,157,157,157,15
    7,157,157,157,157,157,157,157,157,157,157,157,157,157,157,2944
70  1010 DATA 157,157,157,157,157,15
    7,157,157,157,157,157,157,157,157,157,157,157,157,157,157,2943
2D  1020 DATA 32,220 255,201,32,
    208 245,96,169 0 32,177,255,32,183,255,2412
F2  1030 DATA 41,128,208,5,169,1
    11 32 147 255 169,0,162,0,169,222,197 2035
39  1040 DATA 32,160,255,232,201
    ,13,208,245,32 174,255,76,212,255 0,50,2357
40  1050 DATA 0,0,0,0 0 0,0,0,0,
    0,0 0 0,0,0,0
00  1060 DATA 0,0,0 0,0,0,0,0,0,
    0,0,0,0 0,0,0,0
D2  1070 DATA 0,0,0,0,0,0,0,0 0 32
    ,18,158 169,73,141,222,197,1051
78  1080 DATA 76 184 197 162,0,1
    69,13,157,224,197 232 224,38
    ,208 249 96 2424
AF  1090 DATA 32,19,158,169 96,1
    41,222,197 76,184,197 162,0,169,100,157,2105
D9  1100 DATA 251 6,232 236 65 3
    ,208 245 162,0,142 64 3 32,157,158,2004
EF  1110 DATA 32,228 255 240,248
    ,201 20,208,6 32 133 198,76,61,198 201 2337
22  1120 DATA 13 208 9 169,100,1
    74,64 3 157 251 6,96,201,91,48,3,1553
0E  1130 DATA 76 61 198 201,32,1
    6,3 76,61 158 201 42,208,3,7 6 61 1513
45  1140 DATA 156,174,64,3,41,63
    ,157,251 6,234,64,3 174 64 3 236 1735
07  1150 DATA 65,3 208,185,96,17
    4,64 3,224 0 208 1 96 202,16 9 100 1758
61  1160 DATA 206,64,3 174 64,3,
    157 251 6 157 252 6 96,174,6 4 3,1680
65  1170 DATA 169 160 157,251,6,
    96 32,18 178 192,192,192,192 ,192 192 192,2408
D5  1180 DATA 192 192 192,192,15
    2,192,192 192 192,192,192,15 2 192 192,192 192 3072
60  1190 DATA 192-192,192,192,15
    2,192 192,192 192 192,192,15 2,192 174,13,32 2715
1A  1200 DATA 18,221,32,32,32,32
    ,32,32,32 32,32 32 32,32,32,32,32,607
A0  1210 DATA 32,32,32 32 32,32,
    32 32 32,32 32,32,32 32 32,3 2 512
AF  1220 DATA 32 32,32 32,32,32,
    221 13,32,18,173,192,192,192 192,192 1609

B7  1230 DATA 192,192,192,192,15
    2,192 192,192,192,192,192,15 2,192,192,192,192,3072
49  1240 DATA 192,192,192,192,15
    2,192,192,192,192,192,192,15 2,192,192,192,192,3069
FF  1250 DATA 157,157,157,157,15
    7 157 157,157,157,157,157,15 7,157 157,157,157,2512
85  1260 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15 7,157,157,157,157,2512
23  1270 DATA 157,157 157 157,14
    5,0,32,18 176,192,192,192,15 2,192 192,192,2943
A4  1280 DATA 192,192,192,192,15
    2,192,192 192 192,192 192 15 2,192,192,192 192 3072
DC  1290 DATA 192 192,192 192 15
    2,192,192 192,192,192,192,15 2,192 174,13,32,2715
A9  1300 DATA 18,221,32,32,32,32
    ,32 32 32 32 32 32,32,32,32 32,687
C4  1310 DATA 32,32,32,32,32,32,
    32,32 32 32 32 32,32 32 3 2 512
77  1320 DATA 32,32,32 32,32,32,
    221,13,32,18 221,32,32,32,32 32,857
90  1330 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32 3 2 512
BC  1340 DATA 32,32,32 32 32,32,
    32,32,32,32,32,32,32,32 32,2 21,701
FE  1350 DATA 13,32,18,173,192,1
    92,192,192,192 192 192,152,1 92 192 192,192,2540
74  1360 DATA 192,192,192,192,15
    2,192,192 192,192 192,192,19 2,192,192,192 192 3072
1B  1370 DATA 192 192 192,192,15
    2,192 192 192 185,157,157,15 7,157 157,157,2824
FE  1380 DATA 157,157,157 157,15
    7,157,157 157,157,157,157,15 7,157,157,157,157,2512
A2  1390 DATA 157,157,157 157,15
    7,157,157,157,157,157,157,15 7,157,145 145 0 2331
F8  1400 DATA 169,17,141,65,3,24
    ,160,0,162,14,32,240,255,169 166,160,1777
20  1410 DATA 198,32,30,171,169,
    60 160 200 32 30 171,32,19,1 98,32,43,1585
FA  1420 DATA 198,173,64,3,240 1
    1 169 83 141,222 197,32,103,200,76,184,2096
A6  1430 DATA 197 76,115,192,208
    ,76,69,65,83,69,32,197,78,84 69 82 1692
78  1440 DATA 32,198,73,76 69 78
    ,65,77,69,32,212,79,32,211 6 7 82 1452
1E  1450 DATA 65,84,67,72,46,13
    0,162 0 169 251 6 201,100,24 0,14,1510
74  1460 DATA 201,31,16 3 24 105
    ,64,157,224,197,232,76,105,2 00,165,13,1017
A5  1470 DATA 157 224,157,96,165
    ,20,141,65,3,24,160 0,162,14 ,32 240,1704
46  1480 DATA 255,169,166 160,18
    0,32 30 171 169 184,160,200, 32,30,171 32,2155
34  1490 DATA 19,169 32,43 198,1
    73,64,3,240 11,169,78,141,22 2 197 32,1020
00  1500 DATA 103,200,76,184,197
    ,76 115,192 197,78,84,69,82, 69,32,33,1702
0F  1510 DATA 40,156,73 83 75 32
    ,206,65,77 69 41 44,73,60,39 ,32,1213
5F  1520 DATA 198,79,82,32 130 7
    9 82 77 65,84,13,0,168,35 14 1 65 1389
5A  1530 DATA 3 24,160,0,162,14,
    32,240,255 169,166 160 198 3 2,30 171,1016
15  1540 DATA 169,16,160,201,32,
    30,171,32 15,196,32,43,198,1 73,64,3 1541
82  1550 DATA 240,11,169 82 141,
    222,197 32 103,200 76,184,19 7,76,115,192 2237
59  1560 DATA 210,69 78,65,77 69
    ,58,32 39,40,206 69 87,32,20 5,65 1402
2A  1570 DATA 77 69 41 61 40 207
    76 68,206,65,77 69 41,39,13 ,0 1149
0E  1580 DATA 32,15,202,165,17 1
    41 66 3,24,160,0,162,14,32,2 40 255 1535
21  1590 DATA 169 166,160 198,32
    ,30,171,165,167,160,201,32,3 0 171 32 19 1507
02  1600 DATA 108 32 43,198,173,
    64 3,240 75 32 103,200,32,12 5,203 165,1880
E3  1610 DATA 122,160,202 32 30
    171 32 207,202 32,215,202,76 243 202 165,2237
A7  1620 DATA 240 160 201,32,30,
    171,32 58 203 169,2 32,196 2 58,32,204,2010
6E  1630 DATA 255,165 62 160,204
    32 30,171,32,207 202,165 32 160 204,32 2121
D9  1640 DATA 30,171 32 110,204,
    32,166,203 32,130,204,32,204 ,255 32 176 2021
8F  1650 DATA 204 76 212 196 76,
    115,192,208 76,69 65 83,69 3 2,197 78 1348
58  1660 DATA 84 69,82 32,158,73
    ,76 69 78,65,77 69 32 84 78 32,1195
AD  1670 DATA 155 78 80 88,46 13
    ,17 32 32 32,32,32,32,32,32 32 207
83  1680 DATA 32,32 32,32,32,32
    32,32,32,32,32,32 32,32,32,3 2 512
64  1690 DATA 32,32 32,32 32 32,
    32,32,32,32,32,32 32 13 0 461
A0  1700 DATA 145,32,32 32,32,32
    ,32,32 210 69,65,60 73,70,71 ,32,1036
35  1710 DATA 46,46,46,32,194 76
    ,79,67,75,32,206 79,46 32,48
```

```
,48,1152
24  1720 DATA 48,13,0,24,160 0,1
    52,14 32 240,255 166 70,160,
    155,32,1578
A2  1730 DATA 30,171,166,44,160,
    202 32 30 171,76,151,193,157
    ,215 65 02 1548
6F  1740 DATA 78,73 78,71 33 32,
    155,75 80 89 32 215,73 76,76
    ,32,1312
C4  1750 DATA 157 32,65 83,86,32
    ,193,78,89 32 208,82 76,71 8
    2 65 1507
A4  1760 DATA 77,13,25,25,10,201
    ,78,32,205 66,77,73,82,89,32
    46 1196
34  1770 DATA 46 45,32,195 73,78
    84 73 78,85,65,32,195,76 80
    ,69,1340
20  1780 DATA 32,40,69 47,78,41,
    32 63 13,0,145,145,145,25,26
    ,25,957
03  1790 DATA 18,208,76,65,67 69
    ,32 210,157,163 166 32,196 7
    3 83 75 1790
BE  1800 DATA 32,201,78,32,195,0
    2,73,86,69 32 40,210,66 84,8
    5 82,1451
60  1810 DATA 78,41 17,17,17,13,
    32 32,32,32,32,32,32,32,32,3
    2,903
C4  1820 DATA 32,32,32,32,32,32,
    32 32,32,32,32,32,32,32,32,3
    2,512
ED  1830 DATA 32,32 32,32,32,32,
    32,32 32,32 32,32,32,13,0,32
    461
1F  1840 DATA 228,255,201,13,208
    245,96,169,2,162,8,160,2,32
    ,186,255 2226
DD  1850 DATA 173 65 3 162,146 1
    60 203 32 189,255,32,192 255
    ,162 2 32,2063
53  1860 DATA 198 255,96 169,15,
    162 8 160 15,32,186,255-169
    0,32 189 1941
67  1870 DATA 255 32,192,255,162
    ,15,32 198,255 32,207 255 20
    1,48 208 23 2370
14  1880 DATA 32 207,255,201,48,
    208,16 196 15,32,195,255,169
    2 32 195 2031
74  1890 DATA 255 32 215 202,76,
    111 201 169 15,32,195,255,16
    6 2 32 165 2156
08  1900 DATA 255 76,212,196 166
    ,1 133 252 169,0,133 253 160
    0,32 207 2256
8C  1910 DATA 255 145 252,230 25
    2 208 16 162 2 32 105,203 23
    0,253 165 253 2763
74  1920 DATA 157 56,208,3,76,21
    5 203 36 144,80 227,185 252,
    133 254,165,2418
84  1930 DATA 253,133,255,162,2,
    32,105 203 56 254 21 7,109,2
    1,7,201,1641
DA  1940 DATA 50,200,6,166,48,15
    7,21,7,202,76,105,203 36 162
    ,0,189,1710
D7  1950 DATA 224,167,157,146,20
    3 201,13,240,5,232,224,16,20
    0,241 142,65,2514
84  1960 DATA 3,96,0 0 0,0 0 0,0
    ,0 0,0 0 0 0 66
84  1970 DATA 0,0,0,0,0,0,0,169,1,
    133,252,166,8,133,253,169,54
    ,1341
38  1980 DATA 133,1,160,0,55 165
    252,228 254 165,253,228,255
    ,176,18,177,2323
OO  1990 DATA 252,32 210,255,230
    252,208 236 162,2 32,156,20
    4,230 253 208 2922
DC  2000 DATA 227 165,55,133,1,1
    92 2 32 156,204,50,24,160,0,
    162,21,1604
FF  2010 DATA 32 240 235 189,186
    ,160 190 32,30,171 166,235,1
    60 203 32 30,2302
BF  2020 DATA 171,32,136,204,32,
    204 255 32,176,157 104 104,7
    6 115 152 10,2050
01  2030 DATA 57 57,44 188 73 76
    69 32 212,78 79 32,204,78,7
    8,71 1440
F8  2040 DATA 32,212,79,32 195,7
    9,80,69,44,48,96,44,48,45,13
    ,0,1101
FO  2050 DATA 32 32 32 32,32,32,
    32,215,82 73 84 73 78,71 32,
    46,978
53  2060 DATA 46,46,32,192,76,79
    ,67,75,32,206,79,46,13,0 145
    ,145,1281
BB  2070 DATA 145 145 145,28,28,
    28,10,208 76,65,67,63,32,215
    210,201,1683
OA  2080 DATA 212,157,32 156,73
    83,75,32,201 78,32,156,92,73
    ,86 69,1717
FB  2090 DATA 32 40,210,69,84,85
    82 78 41,17,17,17,13,0,169,
    1 955
66  2100 DATA 162,8,160,1 32,186
    255 173 65 3 162,146 160,20
    3 32 189,1837
83  2110 DATA 255,32,192,255,162
    ,1,32,201,255 96,169,2,32,16
    5,255,169,2303
AO  2120 DATA 1,32,195,255 166,1
    5,32 195,255 76 204 255,222,
    21,7,185 2123
D8  2130 DATA 21,7,201,47,208 6,
    169,57,157,21,7,202,76 156,2
    04,96,1638
7C  2140 DATA 162 0,169,0 157,0,
    8 232 224,40,208,245,66 169,
    155,160 2070
F7  2150 DATA 204,32,30,171,78,1
    12,206 15 17 17,25 26,26,25,
    26 29 1057
81  2160 DATA 25,25,26,26,26,29,29,
    29,29,26,26,28,25,25,29,29,1
    0,453
D4  2170 DATA 215,192,192,192,15
    2,132,192,192 192,178,13,26,
    29,29,29,29,2092
C5  2180 DATA 26 25,25,25,25,26,
    29,29,29,25,25,29,29,29,29,2
    9,464
A5  2190 DATA 19,221,165,79 76,7
    6,85,82,83,32,221,13,26,26,2
    9,28,1300
F9  2200 DATA 26,25,25 25 25 26,
    28,29,29 29,26 26,26,25,28,2
    9,464
B3  2210 DATA 28,18,221,195 85 7
    7,80,32,35,52,32,221 13 26 2
    9,25 1178
BO  2220 DATA 26,25,25,25,29,26,
    25,25,25 25,26,25,25,25 25,2
    9,464
86  2230 DATA 25,26,18 221,205,8
    5 73 84,32,32,32,32,221,13,2
    6,26,1188
B1  2240 DATA 29,25,26,26,26 25
    26,29,29,29,26,25 25 25 25,2
    9,464
68  2250 DATA 25 29,25 18,173,16
    2,192,192,192 162,162,162,19
    2,109,13,0,2016
62  2260 DATA 32 228,255 240,251
    201 81,208 3 76 115,192,201
    66,208,3,2362
83  2270 DATA 76 138 205,201,67,
    208,233,76,242,205 169,1,133
    ,255,32,195,2440
SF  2280 DATA 192,166,4 133 186,
    169,126,133,184,169,0 162 4,
    133,113,134 2011
F9  2290 DATA 114 133,183 133,18
    5,32,192,255,166 184 32 201,
    255,162 25 169 2421
EE  2300 DATA 13,32,210,255,32,2
    25,255,240,46 160,0 177,113
    133,103,41 2035
5B  2310 DATA 63,6,103,36 103 16
    ,2,8,129 112 2 9,64,32,210 2
    55,1150
62  2320 DATA 200,192 40 208 230
    ,152,24,101,113 133 113 144,
    2,230,114,202,2194
FE  2330 DATA 208 205,169 13,32,
    210 255,32,204 255,162 125 3
    2,189 255 76,2428
A4  2340 DATA 115 192,169,3,160,
    208 32,30,171 169 103 160,20
    6,32 30 171,1855
BD  2350 DATA 76 14 207 18,17,17
    ,17,17,17 17,17,13,32,32,32
    32,576
A3  2360 DATA 32 32 48,32 46 46
    46 32,154 76,65 67,75 32 32
    32,807
23  2370 DATA 56 32 46 46 32,207
    82 65,78,71 56 13 32 32,32,
    32,525
35  2380 DATA 32,32,48,32 46 46
    46,32,215,72 73 84 69 32 32,
    32,824
CF  2390 DATA 57,32 46,46 46 32,
    154 82,75 87 70 13,32,32,32
    32,520
76  2400 DATA 32 32,50,32,46,46,
    46,46 46,32 210,69,60 32 32,
    32,851
05  2410 DATA 65,32 46 46 46,32,
    210 66,58,32 50 13 0 32 32,3
    2 805
```

```
CF  2420 DATA 32 32,32,51,32,48,
    46,46,46 32 195,85,85,78,32
    32,888
1B  2430 DATA 32,88,32,48,48 32,
    195,82,69,89,32,48,13 32,32,
    32 883
8D  2440 DATA 32,32 32,52,32,46,
    46 32,200,85 82,80,76,69,32,
    32,588
9B  2450 DATA 32,67 32 46,46 32,
    195 82 69,09,32 50,13,32,32,
    32,555
21  2460 DATA 32 32 32,53,32 46
    48 46,32 195 82,69,69,70,32,
    32,912
F3  2470 DATA 32,88,32,48,32,195
    ,82 69,69,78 32,50,13 32,32,
    32,859
E5  2480 DATA 32,32,32,54,32 48
    46 46 48 32 154,76,85 69,32,
    32 886
77  2490 DATA 32,85,32,48,48,32,
    154,76 85,69 32,50,13,32,32,
    32,872
E0  2500 DATA 32 32,32,55,32,48
    48,32 217,65,78,76,78,87,32,
    32,975
01  2510 DATA 32,70,32,46,46,32,
    195,82 85 85 32,51,13,0,185,
    56,1018
E5  2520 DATA 160 207 32 30 171,
    32,170,207,141,32,208,185,54
    ,160 207,32 2052
10  2530 DATA 30,171 32,170,207,
    141 33 208 185,132,160 207,3
    2,30 171,32,1925
1F  2540 DATA 170,207,141,134,2,
    78,115,192,13,17,32,32,212,8
    9,60,69,1501
46  2550 DATA 32 195,79,88 85 32
    ,195 79,82,32,194,79,82,60 6
    9,82 1440
A5  2560 DATA 32,195,79,76,79,85
    ,82 48,48 48,48,48,32,0,13,1
    7,320
53  2570 DATA 32,32,212,85,80,85
    ,32,195,79,66 69,32 196,79,8
    2,32 1300
28  2580 DATA 194,85,67,75,71,78
    ,68 32 195,79,76,79,85,82,48
    ,48,1338
34  2590 DATA 48 48,32,0 13 17,3
    2,32,212,85,80,69,32,195 79,
    68,1042
C5  2600 DATA 69,32,198 78,82,32
    ,212,69,88,84 32 195 78,76,7
    9,85,1491
1C  2610 DATA 82,46,46,48,48,46,
    46,46,32,0,32 228 255,240,26
    1,201,1843
97  2620 DATA 71,18,247,201 48 4
    8,243,201,85,16,4,32,210,255
    ,96,32 1705
1E  2630 DATA 210,255,96,233 55,
    96,0,0 0 0,0 0 0,0,0,0 805
79  2640 POKE193,0 POKE194,192 P
    OKE174,198 POKE175 207
28  2650 POKE187,7 POKE188,8 POK
    E183,8 POKE186,8 REM SUBTIT
    UTE POKE 188,1 FOR TAPE
99  2660 POKE185,0
00  2670 SYS2554
```

DON'T MISS OUT

Fill in your name and address and give this form to your newsagent

Please order me a copy of YOUR COMMODORE and reserve/deliver me a copy every month

NAME .....

ADDRESS .....

.....

.....

.....

Newsagent This magazine is made available to your wholesaler through
S M Distribution Ltd
6 Leigham Court Road
Streatham
LONDON
SW16 2PG

Tel 01-677 8111

Mailing List 128

Address your Christmas cards in plenty of time with our useful address database

Mailing List 128 uses the 40 column screen in 128 mode and produces a handy database of names and addresses on cassette or disk When you have entered all the addresses in the file, they can be printed out onto labels that should be available from your local computer dealer

Your address file doesn't have to be printed, you can use it just to keep all of your friends addresses together in one place When you enter the addresses you will be asked for a telephone number which can be entered if you're using them for reference but if you wish to have your addresses printed out, you can either enter part of the address in this space or leave it blank When you're entering addresses it isn't necessary to put in commas and full stops because these are all automatically inserted.

Enter

There are five lines of data that can be entered including the name and telephone number (if required). Up to 1000 addresses can be held in one disk or cassette file and the number of each address is displayed at the top of the screen along with the maximum number of addresses.

Read

All of the addresses can be viewed on the screen in order of entry or, if your looking for a particular person's address, you can just enter a name and the relevant address will be shown

As you scan through the file, the current address number and total number of addresses in memory are shown at the top of the screen

Print

When you are ready to print your addresses, position the printer head about 5mm from the top of the first label and press F1 All of the addresses will then be printed in order of entry

Erase

When you wish to erase an address you can either scan through the addresses and erase them as you go along or you can use the MATCH NAME option where you just enter the name and the entry will be erased

Load and save

All your addresses are saved into a sequential file named 128 MAIL LIST If you are using a disk drive, the disk status is shown in the form of 00,ERROR MESSAGE 00 All of the error types are explained in the Technical Information section of this Guide

Change

If an address or telephone number changes, the file can be updated by scanning through the addresses and changing them as they appear on the screen by using the MATCH option

You can easily change any line of the address by entering the line number and typing in the new information As you change the address its new form is shown at the top of the screen

Exit

When this option is selected you are asked whether you have saved your new addresses If not, type N and you will be taken back to the main menu.

```
 PROGRAM  MAILING LIST 128

10 REM  ************************
**********
30 REM  *          MAILING LIST 1
28       *
70 REM  *          BY JOHN BOSWOR
TH       *
90 REM  ************************
**********
100 REM ** MAILING LIST PROGRA
M **
```

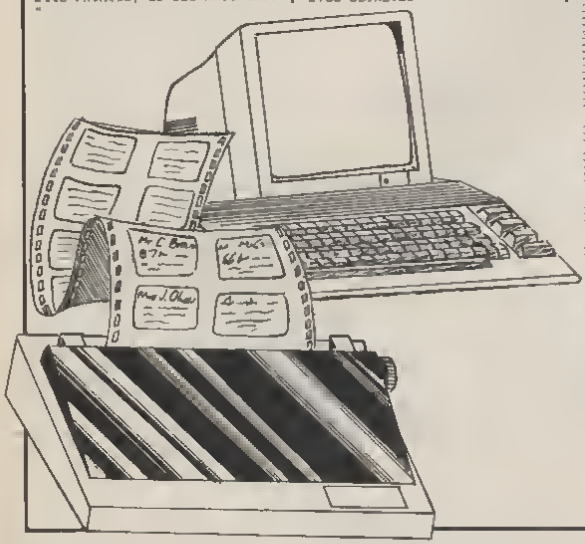
```
110 KEY1,"1":KEY2,"2" KEY3 "3"
 KEY4,"4"
120 KEY5 "5" KEY6,"6":KEY7,"7"
 KEY8,"8"
130 X=1001 DIMNS(X),AE(X),DS(X
),QE(X),PS(X):N=0
140 COLOR0,2 COLOR4,5 COLOR5,1
150 PRINTCHR$(8) GRAPHIC1 1
160 COLOR1,1:PRINT CHAR1,12,2,
"  MAILING LIST  ",1
170 BOX1,20,50,120,130 BOX1,17
,47,117,127 BOX1,14,44,114,124
180 BOX1 185,100,270,100 BOX1,
```

```
188,97,287,177 BOX1,159 54 264
,174 BOX1,5 5 314 195
190 CHAR1 18,4    128    ,1
200 CHAR1,3 0,"F1 - ENTER"
210 CHAR1,3,10,"F2 - READ"
220 CHAR1,3,12,"F3 - LOAD
230 CHAR1,3,14,"F4 - SAVE"
240 CHAR1,21,14,"F5 - PRINT
250 CHAR1,21,12,"F6 - CHANGE"
260 CHAR1,21,10,"F7 - ERASE"
270 CHAR1,21,24,"F8 - EXIT"
280 CHAR1,25,6,"BY"
290 CHAR1,20 8  JOHN BOSWORTH"
```

by J. Bosworth

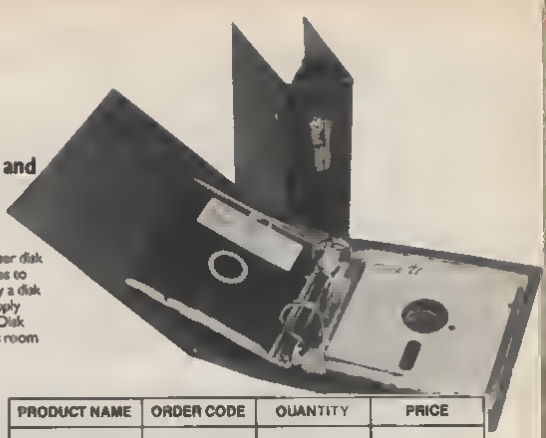
```
300 GETKEYC$
310 IFC$="1"THENGOSUB400
320 IFC$="2"THENDOSUB720
330 IFC$="3"THENDOSUB1920
340 IFC$="4"THENGOSUB2280
350 IFC$="5"THENDOSUB1140
360 IFC$="6"THENGOSUB2580
370 IFC$="7"THENGOSUB1360
380 IFC$="8"THENGOSUB3190
390 GOTO150
400 REM ** ENTER **
410 GRAPHIC0,1
420 IFN>=X-1THEN RETURN
430 CHAR1,2,2,"F1 - CONTINUE"
440 CHAR1,2,4,"F3 - RETURN TO MENU"
450 GETKEYC$
460 IFC$="1"THEN490
470 IFC$="3"THENGOTO390
480 GOTO450
490 N=N+1
500 SCNCLR
510 CHAR1,10,2,"ENTER NAMES/ADDRESSES ,1
520 PRINT PRINT"ENTRY #",N "OF ",X-1
530 INPUT"NAME",NS(N)
540 LETN$(N)=NS(N)+","
550 INPUT"ADDRESS 1",A$(N)
560 LETA$(N)=A$(N)+","
570 INPUT"ADDRESS 2",D$(N)
580 LETD$(N)=D$(N)+","
590 INPUT"ADDRESS 3",V$(N)
600 LETV$(N)=V$(N)+","
610 INPUT"TEL NO - ",P$(N)
620 LETP$(N)=P$(N)+" "
630 IFD$(N)=""THENLETD$(N)="-"
640 IFV$(N)=""THENLETV$(N)="-"
650 IFP$(N)=""THENLETP$(N)="-"
660 PRINT PRINTCHR$(18)"ANOTHER ENTRY? (Y/N)"
670 GETKEYC$
680 IFN>=X-1 THEN RETURN
690 IFC$="Y"THEN GOTO 490
700 IFC$="N"THEN RETURN
710 GOTO 670
720 REM ** READ **
730 GRAPHIC0,1
740 IFN=0 THEN RETURN
750 CHAR1,10,2," READ NAMES/ADDRESSES ,1
760 CHAR1,2 4,"F1 - READ ALL"
770 CHAR1,2,6,"F3 - MATCH NAME "
780 GETKEYC$
790 IFC$="1"THEN820
800 IFC$="3"THEN540
810 GOTO780
820 FOR1=1TON
830 SCNCLR
840 PRINT PRINT" ENTRY NO ",I OF ,N
850 PRINT PRINT" NAME " N$(I)
860 PRINT" ADDRESS 1 ",A$(I)
870 PRINT" ADDRESS 2 ",D$(I)
880 PRINT" ADDRESS 3 " V$(I)
890 PRINT" TEL.NO - ",P$(I)
900 PRINT PRINTCHR$(18)"PRESS ANY KEY TO CONTINUE"CHR$(146)
910 GETKEYC$
920 NEXTI
930 RETURN
940 SCNCLR
950 PRINT PRINTAB(2)"MATCH WHICH NAME", INPUT$
960 LET$=I$+","
970 FORI=1TON
980 IFN$(I)=T$THENGOTO1050
990 NEXTI
1000 PRINT PRINTCHR$(18)"NO SUCH NAME IN FILE"CHR$(146)
1010 PRINT PRINT"PRESS ANY KEY TO RETURN TO MAIN MENU"
1020 GETC$ IFC$=""THEN1020
1030 RETURN
1040 SCNCLR
1050 PRINT PRINT" ENTRY NO " 1 ,"OF",I
1060 PRINT PRINT" NAME ",N$(I)
1070 PSINT" ADDSESS 1 ",A$(1)
1080 PRINT" ADDRESS 2 ",D$(I)
1090 PRINT" ADDRESS 3 ",V$(I)
1100 PRINT" TEL NO - ",P$(I)
1110 PRINT PSINTCHR$(18)"PRESS ANY KEY TO RETURN TO MAIN MENU"CHR$(146)
1120 GETC$ IFC$=""THEN1120
1130 RETURN
1140 REM ** PRINT **
1150 GRAPHIC0,1
1160 CHAR1,2,2,"F1 - CONTINUE"
1170 CHAR1,2,4,"F3 - RETURN TO MENU"
1180 GETKEYC$
1190 IFC$="1"THEN1220
1200 IFC$="3"THEN RETURN
1210 GOTO 1140
1220 SCNCLR PRINT PSINTCHR$(18)" PRINTING ADDRESSES "CHR$(146)
1230 PRINT PRINTCHR$(18)" PLEASE WAIT "CHR$(146)
1240 OPEN1,4
1250 FORI=1TON
1260 PRINT#4,N$(1)
1270 PRINT#4,A$(I)
1280 PRINT#4 D$(1)
1290 PRINT#4,V$(I)
1300 PRINT#4,P$(1)
1310 PRINT#4
1320 PRINT#4 PRINT#4
1330 NEXTI
1340 CLOSE1
1350 RETURN
1360 REM ** ERASE **
1370 IFN=0 THEN RETURN
1380 GRAPHIC0,1
1390 CHAR1,2,2,"F1 - CONTINUE"
1400 CHAR1,2,4,"F3 - RETURN TO MENU"
1410 GETKEYC$
1420 IFC$="1"THEN1450
1430 IFC$="3"THENRETURN
1440 GOTO1360
1450 SCNCLR
1460 CHAR1,10,2,"ERASE NAMES/ADDRESSES",1
1470 CHAR1,2,4,"F1 - SCAN ALL"
1480 CHAR1,2,6,"F3 - MATCH NAME"
1490 GETC$ IFC$=""THEN1490
1500 IFC$="1"THENI530
1510 IFC$="3"THEN1700
1520 GOTO1360
1530 FORI=1TON
1540 SCNCLR
1550 PRINT"ENTRY #",I "OF",N
1560 PRINT'NAME ",N$(I)
1570 PRINT ADDRESS 1 A$(I)
1580 PRINT ADDRESS 2,",D$(I)
1590 PRINT ADDRESS 3 ",V$(I)
1600 PRINT TEL NO - ",P$(I)
1610 PRINT PRINT F1 - CONTINUE

1620 PRINT PRINT"F3 - ERASE"
1630 GETKEYC$
1640 IFC$="1"THEN GOTO 1660
1650 IFC$="3"THEN GOSUB 1830 DOTO1970
1660 GOTO1630
1670 N=N-1 I=I-I
1680 NEXTI
1690 RETURN
1700 SCNCL$
1710 INPUT"MATCH WHICH NAME',T$
1720 LET1$=I$+" '
1730 FORI=1TON
1740 IFN$(I)=I$ THEN DOSUB 1830
1750 NEXTI
1760 PSINT"NO SUCH NAME IN FILE"
1770 PRINT"PRESS ANY KEY TO RETURN TO MENU"
1780 GETKEYC$ GOTO1620
1790 RETURN
1800 PRINT,PRINTCHR$(18)"NO SUCH NAME IN FILE"CHR$(146)
1810 PRINT PRINT"PRESS ANY KEY TO RETURN TO MENU"
1820 GETC$ IFC$=""THEN1820
1830 FORK=1TON
1840 N$(K)=N$(K+1)
1850 A$(K)=A$(K+1)
1860 D$(K)=D$(K+1)
1870 V$(K)=V$(K+1)
1880 P$(K)=P$(K+1)
1890 LETN=N-I LETI=I-I
1900 NEXTK
1910 GOTO150
1920 REM ** LOAD **
1930 GRAPHIC0,1
1940 CHAR1,2,2,"F1 - CONTINUE"
1950 CHAR1,2,4,"F3 - RETURN TO MENU"
1960 GETKEYC$
1970 IFC$="1"THEN2000
1980 IFC$="3"THEN RETURN
1990 GOTO1920
2000 SCNCLR
2010 CHAR1,12,2," LOAD FILE ",1
2020 CHAR1 2,4,"F1 LOAD FROM DISK"
2030 CHAR1,2,6, F3 LOND FROM TAPE"
2040 GETKEYC$
2050 IFC$="1"THEN2080
2060 IFC$="3'THEN2240
2070 GOTO2040
2080 OPEN1 8,3,"0 10$ MAIL LIST,SEQ"
2090 INPUT#1,N
2100 FORI=1TON
2110 INPUT#I,N$(I)
2120 LETN$(I)=N$(I)+","
2130 INPUT#I,A$(I)
2140 LETA$(I)=A$(I)+ ,"
2150 INPUT#1,D$(I)
```

```
2160 LET D$(I)=D$(I)+","
2170 INPUT#I,U$(I)
2180 LET U$(I)=U$(I)+","
2190 INPUT#I P$(I)
2200 NEXTI
2210 CLOSE1
2220 IFC$="1"THENPRINT,PRINT P
RINT'  DISK STATUS - "DS$ SLEE
P3
2230 RETURN
2240 PRINT"", OPEN1 1 8 120 M
AIL LIST
2250 GOTO2050
2260 REM ** SAVE **
2270 IFN=0THEN RETURN
2280 GRAPHIC0,1
2290 CHAR1,2,2,"P1 - CONTINUE"
2300 CHAR1,2,4,"P3 - RETURN TO
 MENU"
2310 GETKEYC$
2320 IFC$="1"THEN2350
2330 IFC$="3"THEN RETURN
2340 GOTO2260
2350 SCNCLR
2360 CHAR1 12,2,"  SAVE FILE
 "+1
2370 CHAR1,2,4,"F1 - SAVE TO D
ISK"
2380 CHAR1,2,6, F3 - SAVE TO T
APE"
2390 GETKEYC$
2400 IFC$="1"THEN2435
2410 IFC$="3 THEN2560
2425 GOTO2390
2430 OPEN1,8,15
2440 PRINT#1,"S0 120 MAIL LIST
"
2450 CLOSE 1
2460 OPEN1,8,3, 0 120 MAIL LIS
T,SEQ,WRITE"
2470 PRINT#1 N
2480 FORI=1TON
2490 PRINT#1 N$(I)
2500 PRINT#I A$(I)
2510 PRINT#1,D$(I)
2520 PRINT#1,U$(I)
2530 PRINT#1 P$(I)
2540 NEXTI
2550 CLOSEI
2560 IFC$="1"THENPRINT PRINT P
RINT"  DISK STATUS - "DS$ SLEE
P3
2570 RETURN
2580 PRINT"", OPEN 1 1,1,"120
MAIL LIST"
2590 GOTO2470
2600 REM ** CHANGE **
2610 IFN=0 THEN RETURN
2620 GRAPHIC0,1
2630 CHAR1 2 2,"F1 - CONTINUE"
2640 CHAR1,2,4,"F3 - RETURN TO
 MENU"
2650 GETKEYC$
2660 IFC$="1"THEN2690
2670 IFC$="3"THENRETURN
2680 GOTO2580
2695 SCNCLR
2700 CHAR1,15,2," CHANGE AN E
NTRY  ",1
2710 CHAR1,2,4 "F1 - SCAN & CH
ANGE"
2725 CHAR1,2,6,"F3 - MATCH"
2730 GETKEYC$
2740 IFC$="1"THEN2770
2750 IFC$="3"THEN2810
2760 GOTO2730
2770 FORI=1TON
2780 SCNCLR PRINT#S(I)
2790 PRINTA$(I)
2800 PRINTD$(I)
2810 PRINTU$(I)
2820 PRINTP$(I)
2830 PRINT PRINT"F1 - CONTINUE
"
2840 PRINT'F3 - CHANGE"
2850 GETKEYC$
2860 IFC$="1"THEN GOTO 2850
2870 IFC$="3"THEN GOSUB 3050 G
OTO2850
2880 GOTO2850
2890 NEXTI
2900 RETURN
2910 SCNCLR
2920 INPUT"MATCH WITH WHAT NAM
E ,M$
2930 LET M$=M$+',"
2940 FORI=1TON
2950 IFN$(I)=M$THENGOSUB 3010
RETURN
2965 NEXTI
2970 PRINTCHR$(18)"NO SUCH NAM
E IN FILE"CHR$(146)
2980 PRINT"PRESS ANY KEY TO RE
TURN TO MAIN MENU'
2990 GETC$ IFC$=""THEN2990
3000 RETURN
3010 SCNCLR PRINTN$(I)
3020 PRINTA$(I)
3030 PRINTD$(I)
3040 PRINTU$(I)
3050 PRINTP$(I)
3065 PRINT"1 >> NAME"
3070 PRINT"2 >> ADDRESS 1"
3080 PRINT"3 >> ADDRESS 2"
3090 PRINT"4 >> ADDRESS 3"
3100 PRINT"5 >> TEL NO"
3110 PRINT INPUT"CHANGE WHICH"
,C
3120 ONC60SUB3140,3155,3160,31
70,3180
3130 RETURN
3140 PRINT INPUT"CHANGE TO WHA
T",Q$ N$(I)=Q$ LET N$(I)=N$(I)+
",", I=I-1 RETURN
3150 PRINT INPUT CHANGE TO WHA
T",Q$ A$(I)=Q$ LET A$(I)=A$(I)+
" ", I=I-1 RETURN
3160 PRINT INPUT"CHANGE TO WHA
T" D$ D$(I)=Q$ LETD$(I)=D$(I)+
", " I=I-1 RETURN
3170 PRINT INPUT'CHANGE TO WHA
T",Q$ U$(I)=D$ LETU$(I)=U$(I)+
", " I=I-1 RETURN
3180 PRINT INPUT CHANGE TO WHA
T",D$ P$(I)=Q$ LETP$(I)=P$(I)+
" " I=I-1 RETURN
3190 REM ****** EXIT ******
3200 GRAPHIC1,1
3210 CHAR1,2,5,"HAVE YOU SAVED
 NEW ADDRESSES ?"
3220 GETKEYC$
3230 IFC$="Y"THENGRAPHIC0,1:EN
D
3240 IFC$="N"THENRETURN ELSE32
00
```



Binders

Organise and protect your disk with Commodore Disk User disk binders and data disks.



Why not keep your Commodore Disk User program collection alongside your magazines in a stylish Disk User disk binder? The binder comes complete with 10 disk sleeves to organise and protect your program disks. Why not buy a disk binder to house all of your data disks? We can even supply Commodore Disk User data disks. The Commodore Disk User logo immediately identifies your disks and there's room to title them and document the disks details
Send for your disks and binders now!

Prices are as follows
Commodore Disk User Binder £4.95, including 10 sleeves Order code **BDYU1**
Commodore Disk User Binder with 10 sleeves and 10 disks, £9.95 Order code **BDYU2**
10 sleeves for insertion in binder, £1.50 Order code **BDSI0**
20 sleeves for inclusion in binder, £2.75 Order code **BDS20**
10 Commodore Disk User data disks, £5.95 Order code **BDDI0**

PRODUCT NAME	ORDER CODE	QUANTITY	PRICE
Overseas postage add £1.00			
CHEQUES PAYABLE TO A.S.P LTD		TOTAL	

All orders should be sent to: YDUR COMMODDRE, READERS SERVICES, ARGUS SPECIALIST PUBLICATIDNS, 9 HALL RDAD, HEMEL HEMPSTEAD, HERTS HP2 7BH.
Please allow 28 days for delivery.

Message Construction Kit

Produce customised scrolling displays with a suite of editor programs

by Frank van Tiggelen

The Message Construction Kit is a useful package which can add a pleasing, dual-line banner effect to most programs It can scroll credits across the screen or add wrap-around instructions As long as there's room for an interrupt, MCK can help

Although the messages are referred to as text, they can include user-defined pictures. In fact, the whole text is redefinable to suit your own particular needs

By raising the start of Basic to $3000 (12288), there is room for a character set from $2000 (8192) upwards and a message can be stored in the space from $0800 to $1FFF (2048–8191) giving room for a string of 6144 characters. The routine which forms the interrupt engine is stored from $C000 to $C166 (49152–49510)

The three MCK editors control text, characters and storage accessed via a master menu which leads to sub-menus for each option

Text Editor (T)

There are six options within the text editor

E – selects the 'text edit' facility which allows the scrolling text to be typed in. At first, the program asks for a starting point which should be somewhere in the text storage range of 2048 to 8191 Entering text is then simply a case of typing in the words, using the DEL key to correct any errors Pressing RETURN recalls the option menu.

V – the View Text option is used to display the text from a given point in memory While the text is being displayed it can be paused with the F7 key or terminated by pressing the spacebar.

This function also allows you to check the text length which can then be stored by entering the value through the 'M' option from the text menu. This tells the program where the text ends and the wrap-around begins

S – reveals the main scrolling demo where the results of your labours can be viewed Press the key and the demo is displayed in fully defined characters across the top two screen lines Pressing the spacebar may cause a panic at first because a warm reset occurs This is normal and entering RUN will restore the program without losing any character or text information

R – if text has been omitted, it can be inserted using the option On entry, the program asks for a start and end address of the block of text to be moved, the destination address must then be entered For the end address you have to know how many characters need to be inserted, this can then be added to the start address to give the end value

This facility can also be used to repeat blocks of text. The limitation is that only 2000 characters can be moved at a time

M – As mentioned before, this is the 'set message length' option that tells the system where the loop starts and ends This can be any value from 256 to 6144 in blocks of 256 characters The number of characters is increased with the '+' key and decreased using '–'

X – Exits from the text menu to the main menu

Character Editor (C)

Up to 128 double height characters can be designed and manipulated to create the building blocks for a scrolling message The screen displays a 16 × 8 grid, a menu of options and a full character set The screen layout can be seen in Diagram 1

The functions can be selected by pressing the specially allocated keys displayed alongside the menu but, by pressing the spacebar, the options can be highlighted with a joystick and the option will he executed when the fire button is pressed

The space character (ASCII 32) is not included in the redefinable set so that the screen does not fill up

with rubbish This also means that spaces can easily be inserted in the text editor by pressing the spacebar as normal

When the editing process is complete, the X option will return the program to Basic but the program can be re-run without losing the defined characters from memory

SELECT CHARACTER – This option moves the cursor to the character set at the bottom of the screen. The character to be edited can be selected by moving the cursor onto the relevant character and then pressing fire

CLEAR CHARACTER – When this option is selected, the current character is erased ready for redefining

REVERSE CHARACTER – All of the on pixels are turned off and vice versa when this option is chosen for the character being edited

FIRE MODE PLOT – The fire mode (FM) is set to unit value and a pixel is set when the fire button is pressed

FIRE MDDE ERASE – This is like the last option but the FM value is set to zero and the fire button is now used to erase a pixel

COPY CHARACTER – If you want to make a slightly altered version of another character from the set, this option allows you to copy it. First, the program asks for the character to be copied and this is selected on the lower display using the joystick Next, the character position to which the copy is made is selected in the same way

GET CHARACTER – After selecting a character for display, this function transfers the information to the editing grid

PRINT CHARACTER – A permanent pixel map can be printed out using this facility

MIRROR X – After selection, the character to be mirrored is laterally reversed

MIRROR Y – This is the same as mirroring in the X direction but the character is inverted instead

5 CHAR PAINT – Five consecutive characters can be designed simultaneously on a special 40 × 16 grid After the first character of the sequence has been selected from the character set display, the screen changes to reveal the new grid Beneath this display is the fire mode (1 or 0) and the poke values of the characters displayed The first character must have a screen poke value of less than 123

The five character frieze can now be created using the '+' and '–' keys to select the fire mode When finished, the characters are transferred to the computer's memory by pressing the spacebar F7 displays the characters on the screen as they will appear in the text and, if they look alright, the main editing screen can be re-entered by pressing 'X'

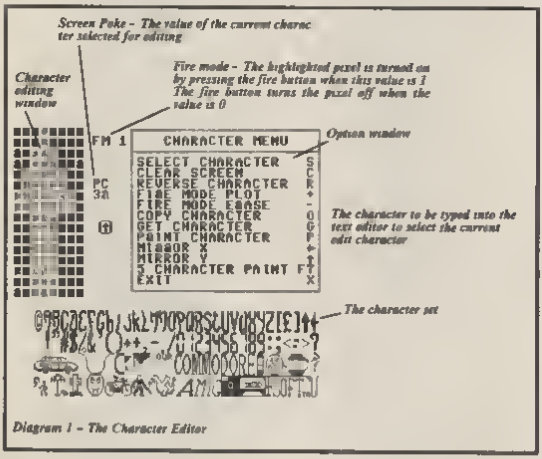


Diagram 1 – The Character Editor

Storage (S)

The options available from the storage menu relate to saving and loading routines

CHANGE DEVICE – The system is initially set for tape operations but this function will toggle from tape to disk

SAVE CHARACTERS – The current character set will be saved under a filename of your own choice

SAVE TEXT – This saves the current text message on to the storage device

SAVE MACHINE CODE – As the scrolling message is modified through the text editor, the machine code is automatically tailored to run the program. This option saves the correct routine to drive the current characters and message

LOAD – Can be used to load any of the three file types saved with the preceding three options

Program creation

To use the files with your own programs, first enter the following

POKE 43,0 POKE 44,48 POKE 12287, 0-NEW

Now the character set, text message and machine code can be loaded in Before loading your own routine type NEW again

The following commands will kickstart the scroller

POKE 899,0 POKE 900,199: POKE 49324,8-SYS 49152

The code routine occupies the locations from 49152 to 49510

```
INSTRUCTIONS FOR ENTERING THE
PROGRAMS
1) TYPE IN AND SAVE 'MESSAGE
   CONSTRUCTION KIT' ONTO YOUR
   MASTER TAPE OR DISK
2) TYPE IN AND SAVE 'M/C
   MAKER' ONTO A SEPARATE
   TAPE/DISK
3) RUN 'M/C MAKER' AND ADD IT
   TO THE MASTER TAPE/DISK
   AFTER 'MCK'
4) TYPE IN AND SAVE 'CHARACTER
   MAKER' ONTO A SEPARATE
   TAPE/DISK
5) RUN 'TEXT MAKER' AND ADD IT
   TO THE MASTER TAPE/DISK
   AFTER 'M/C MAKER'
6) TYPE IN AND SAVE 'TEXT
   MAKER' ONTO A SEPARATE
   TAPE/DISK
7) RUN 'TEXT MAKER' AND ADD IT
   TO THE MASTER TAPE/OISK
   AFTER 'CHARACTER MAKER'
8) LOAD AND RUN 'MCK' AND
   LOAD THE PROGRAMS IN THIS
   ORDER
          M/C
          CHARS
          TEXT
```

PROGRAM: MESSAGE CONSTRUCTION KIT

```
4A  1■ DE=1 DV=1 DS()="TAPE" DA
    (2)="DISK" GOTO1230
3A  2■ POKE53280,12 POKE53281,15
    POKE■■ ■ PRINT"{CM,CLR,BLA
    CK}" POKE53269,1:POKE53287,
    ■
7C  30 FORX=0TO63 POKE832+X,■:NE
    XT POKE834,255 FORX=2TO5 POK
    E832+X*3 155 NEXT J=1
6E  40 POKE853 255 POKE856,255 P
    OKE835 255 POKE53248 24:POKE
    53248,58 POKE2■48 13
05  50 FORX=0TO5:PRINT"{WHITE,R
    VSON SOM}" NEXT PRINT"{HOME,
    DOWN}"TAB(8)"{CM}FM {WHITE}1
    ■ MD=11 GOSUB310
0E  A0 A=■ XX=58 SY$43392 FORD=0
    *32TOO=7 POKE8152+D,0 NEXT P
    RINT"{HOME,DOWN5}"TAB(9)"PC
    ■■
A0  70 PRINTTAB(10)"{DOWN2,CS,CW
    }"■PRINTTAB(5)"{CM,C4■{CS,C
    G}"■PRINTTAB(10)"{CT}":GOTO5
    ■■
D5  ■■ P=PEEK(56320):IF(PAND1)=■
    AND>0THENY=Y-1
02  ■■ IF(PAND2)=■AND>15THENY=Y
    +1
22  100 IF(PAND4)=■AND<0THENX=X
    -1
0E  110 IF(PAND8)=■AND<7THENX=X
    +1
02  1E■ IF(PAND10)=■THENPOKE5529
    6+X+Y*40,MD ■OTO200
24  130 POKE53248,24+X*■ POKE532
    45,58+Y*■:GETA■ IFA$=""THEN■
    ■
■■  140 IFA$="+"THENMD=11:PRINT"
    {HOME,DOWN}"TAB(12)"{WHITE}1
    ■ GOTO■■
40  150 IFA$="-"THENMD=1 PRINT"{
    HOME,DOWN}"TAB(12)"{WHITE}■"
    :GOTO■■
■■  160 IFA$="S"THEN450
32  170 IFA$="C"THENPRINT"{HOME}
    ";:FORD=1TO16 PRINT"{WHITE,R
    VSON,SOM}" NEXT GOTO■■■

3E  180 IFA$="R"THENGOTO■40
45  190 IFA$=" "THEN■A■
E■  200 IFA$="X"THEN510
78  210 IFA$="O"THEN■6■
A5  220 IFA$="■"THEN540
F3  230 IFA$="P"THEN1010
E3  240 IFA$="{F2}"THEN2■■■
0T  250 IFA$="_"THEN■OTO2■2■
F0  260 IFA$="~"THENGOTO2350
■4  270 GOTO6■■
1S  280 ZZ=8192+■*(32*D+C+A*24+A
    *1)+Y IFY>7THEN2Z=ZZ+12■■
30  290 IFMD<>1THENPOKEZZ PEEK(Z
    Z)OR(■^(7-X)) GOTO130
D5  300 POKEZZ,PEEK(ZZ)AND(255-2
    ^(7-X)) GOTO130
13  310 PRINT"{HOME,C5}"TAB(15)"
    {CA 5C2■,C5}"
29  320 PRINTTAB(13)"{5B,5PC3 C4
    }CHARACTER MENUCCS 5PC3 5B}"
    
30  330 PRINTTAB(13)"{C5,CQ,5C2■
    CW}" FORA=1TO12
B5  340 PRINTTAB(13)"{5B}"TAB(34
    )"{5B}" NEXT PRINTTAB(13)"{C
    2,5C2■,CX,C4}"
■D  360 PRINT"{HOME DOWN2}" POKE
    53284,■ POKE53271,■
0T  360 FORX=1TO12 READA$ PRINTI
    AB(14)LEFT$(A$ 10)"{WHITE}"R
    1■HTS(A$,2)"{C4}" NEXT PRINT
    "{HOME}";
BC  370 FORY=■TO31
3■  3■■ POKE17A8+X,X:POKE17A8+X,
    X+32 POKE18G8+X,X+64 POKE)54
    8+X X=56 Y=X
■■  350 X=X-120 POKE17A8+Y,X POK
    E182■+Y,X+32 POKE)588+Y X+54
    POKE)568+Y,X+3G
0C  4■■ X=X-120 NEXT X=■ Y=■ RET
    URN
S7  41■ DATA"SELECT CHARACTER
    S","CLEAR SCREEN(SPCS) C","
    REVERSE CHARACTER: R"
A■  420 DATA"FIRE MODE PLOT(SPC3
    ): ●","FIRE MODE ERASE : -"
    ,"COPY CHARACTER(SPC3): O"
A■  430 DATA"SET CHARACTER(SPC4)
    :P","PRINT CHARACTER P".
    "MIRROR X(SPCS) "
CF  440 DATA"MIRROR Y(SPCS) ",
    "5 CHARACTER PAINT F7","EX1T
    (SPC13) X"
FF  450 F=X:G=Y A=E Y=D:X=C FOX
    E53271,1:POKE53264,A
F7  460 {FPEEK(■■32■)=111THEN■■■
    
1■  470 P=PEEK(56320) IF(PAND1)=
    ■AND>0THENY=Y-1
A■  480 IF(PAND2)=■AND<3THENY=Y
    +1
E2  490 IF(PAND4)=■ANDX=■AND■=1T
    HEN670
■1  500 IF(PAND4)=■AND>■THENX=X
    -1
CC  510 IP(PAND■)=■AND<24THEN■■■
    
■■  520 IP(PAND■■)=■AND<36THENX=
    X+1
PE  530 IF(PAND1■)=■THEN560
2A  540 IPA=1ANDX=7THENX=X-1
73  550 POKE53248,XX+X*■ POKE532
    49,185+■*Y ■OTO470
SE  ■■■ W=8*24+8*1+C+D*■■ POKE55
    ■■■+W,11 POKE55■■■+W,11 W
    =A*24+A*1+C+Y*■■
E1  570 POKE55■■■+W,■ POKE55■■■+
    W+■,■ C=X D=Y E=A X=F Y=G P
    OKE53271,■
1■  580 QQ=(D*32+C+A*24+A*1) PRI
    NT"{HOME,DOWN■,WHITE}"TAB(■)
    ■Q"{LEFT} " :IFDQ<1■THENPRIN
    T" "
CC  590 POKE1■24+1■+8*■,QQ PRIN
    T"{HOME}";:FORL=1TO1■ PRINT"
    {WHITE,RVSON SOM}";NEXT
E■  ■■■ POKE53284,■:IFWQ=1THENRE
    TURN
P3  ■1■ P=PEEK(■■32■) IFP=111THE
    N61■
2T  62■ GOTO■■
91  ■3■ {FDV=1THENDV=2:DN=■:GOTO
    55■
■3  5■■ DV=1 DN=DV
■■  ■■■ PRINT"{HOME WHITE}"TAB(2
    5)DS(DV) GOTO■■
■■  ■■■ X=■ XX=■:A=1:POKE53284,A
    GOTO54■
9E  ■7■ X=24 XX=56 A=■ POKE53284
    ,A GOTO54■
2■  ■■■ POKE5324■,24:POKE53284,1
    :POKE53277 1 C=Y Y=J
F■  63■ P=PEEK(■■32■) IF(PAND1)=
    ■ANDY>1THENY=Y-1
■7  7■■ IF(PAND2)=■ANDY<12THENY=
    Y+1
D5  71■ IF(PAND1■)=■THEN73■
3■  72■ POKE53251,■ POKE53272,■
    
AA  73■ POKE53251,■ POKE53272,■
53  74■ IFWQ=1THENRETURN
A4  75■ P=PEEK(5632■) IFP=111THE
    N75■
D5  76■ J=Y Y=1:ONJGOTO45■ 77■,7
    A■ 73■,■■ ■■■ 5■■ 1■7■ 2■2■
    ,2■A■ 2■■■ ■)■
B5  77■ A$="C" GOTO14■
D■  78■ A$="F" GOTO14■
A5  7■■ A$="●":GOTO14■
■5  8■■ A$="-" GOTO14■
75  81■ SYS65126
27  8C■ FORD=■TO7 ZZ=8192+8*(D*3
    2+C+A*24+A*1)+Q POKEZZ,■:POK
    EZZ+(12■*8) ■:NEXT
3■  83■ GOTO8■
4■  85■ FORD=■TO7 ZZ=8192+8*(D*3
    2+C+A*24+A*1)+Q POKEZZ,255 P
    EEK(ZZ)
SP  85■ POKEZZ+128*■,255-PEEK(ZZ
    +128*■) NEXT GOTO8■
E■  ■■■ WD=1 PRINT"{HOME} ;FORX=
    1TO15:PRINT NEXT PRINT"{BLAC
    K}COPY FROM CHARACTER" GOSUB
    45■
2■  87■ Z1=8192+8*(D*32+C+A*24+A
    *1)
■■  ■■■ PRINT"{HOME,BLACK}";FORK
    =1TO1■:PRINT NEXT
44  ■■■ PRINT"{UP}COPY TO CHARAC
    TER " DOSU■45■:ZZ=8152+8*(D
    *32+C+A*24+A*1)
AC  ■■■ W1=21+128*■ W2=22+128*■:
    FORQ=■TO7
1■  51■ POKEZZ+Q,PEEK(Z1+Q)
5E  52■ POKEW2+Q,PEEK(W1+Q):NEXT
    PRINT"{HOME}" FORK=1TO15:PR
    INT NEXT WD=■ X=■
■■  53■ PRINT"{5PC35}":GOTO5■
24  54■ POKE53280,■
■■  ■■■ ZZ=8192+8*(D*32+C+A*24+A
    *1):WW=ZZ+128*■ FORQ=■TO7:T=
    PEEK(ZZ+Q) S=PEEK(WW+Q)
1■  ■■■ FORR=7TO■STEP-1 IFT=2^R>
    =■THENT=T-2^R POKES■256+Q*■+
    ■(7-R),11 ■OTO3■■
S1  57■ POKE55256+Q*■+7-R,1
29  5■■ {F5-2 R>■THENS=S-2^R:PO
    KES5518+Q*■+7-R 11:GOTO1■■■
    
E1  59■ POKES5518+Q*■+7-R,1
F■  1■■■ NEXT NEXT:POKE53280 11:
    ■OTO8■■
```

```
AE 1010 PRINT"{HOME}";FORK=1TO1
   5 PRINT NEXT PRINT"{BLACK}DO
   YOU OWN A C1T12EN 10P-S40?
   1Y/N1'
80 1020 DETAS 1FA$<>"Y"ANDA$<>"
   N THEN1022
FD 1030 1FA$="Y"THENOPENS,4,6 P
   RINT#6,CHR$(8) CLOSE6
E0 1040 PRINT"CUP SPC3&2"
0A 1050 OPEN4,4 POKE53265 PEEK(1
   53265)AND23
A4 1000 22=8:190=0*10^32+C1 FORD
   =0TO7 T=PEEK(22+D)
FE 1070 FORR=7TO0STEP-1 1FT-2^R
   >=8THENT=T-2^R:PRINT#4,"{RVS
   ON}1 {RVSOFF}",;:GOTO1050
C0 1000 PRINT#4,"{90}";
27 1000 NEXT PRINT#4,"{CD}" NEX
   T
A0 110R ZZ=0:1R2=0*10^32+C}+128*
   0 FORD=0TO7 T=PEEX(ZZ+D)
9F 111E FORR=7TO0STEP-1 1FT-2^R
   >=8THENT=T-2^R PRINT#4 "{RVS
   ON} {RVSOFF}" GOTO1130
20 1120 PRINT#4 "{90}";
2A 1130 NEXT PRINT#4 "{CB}" NEX
   T PRINT#4 "{CT10}" CLOSE4 POK
   E53265,PEEK(53265)OR16
FD 1140 X=8 GOTO88
1F 1150 PRINT"{CLR DOWN2 C41"TA
   8{12I"SAVE "N8 PRINTTAB(12)"
   CCTI61"
44 1160 INPUT"{DOWN2 C516}IVE F1
   LENAME ",F1$ 1FF1$=""ANDDE=0
   THENPRINT"CUP31", GOTO1160
82 1170 POKE154 SA/256 POKE153
   SA-PEEX(154)*200
31 1180 POKE175 FA/200:POKE174
   FA-PEEK(175)*256
53 1190 L=LEN(F1$)
70 1200 FOR1=1TOL POKE1023+1,AS
   C(M1D$(F1$,1 1)) 111 NEXT
4D 1210 POKE167 0 POKE188,4 POK
   E183 L POKE185 DE SYS62554
46 1220 RETURN
8C 1230 PRINT"{CLR DOWN}" POKES
   32B8 0 POKE53281,8
DB 1240 PRINT"{WHITE SPC6}IKE M
   ESSAGE CONSTRUCTION KIT"
CA 1245 PRINT"{WHITE,SPC6 CT28,
   DOWN}
0A 1250 PRINT CC8 SPC41PROGRAMM
   ED BY FRANK VAN TIGGELEN"
21 1260 PRINT"{C4 SPC15}MARCH 1
   988"
97 1270 PRINT"{CS SPC9}FOR THE
   YOUR COMMODORE"
BB 1275 PRINT"{YELLOW SPC10}SER
   1OUS USER S GU10ECCS,DOWN3J"
35 1280 PRINTTA01S1"{CA 5*20,CS
   }
E5 12R0 PRINTTAB15]"{S-} M A 1
   NCSPC4}M E N U {S-}"
E1 1300 PRINTTAM151"{CQ,5*20,CW
   }
E4 1310 PRINTTA01S]"{S-3IEXT ED
   ITOR{SPC6} {WH1E}IC{S,8-}"
03 1320 PRINTTA01S1"{S-}CHARACT
   ER EDITOR  {WH1E}JCCS S-}"
E2 1330 PRINTTA0{R}"{S-}SIORAGE
   {5PC10}  {WH1E}3SCCS S-}"
E5 1340 PRINTTA0IR}"{CQ 5*20 CW
   }"
E0 1350 PRINTTA01S1"{S-}EXIT{SP
   C13}  {WH1E}XCCS,8-}"
D9 1360 PRINTTA01S1"CC2,5*20 CX
   }"
D0 1370 GETA8 1FA8=""THEN137B
7E 1380 1FA8="I"THEN1720
BB 1350 1FA$="C"THEN820
21 1400 1FA$="S"THEN1430
6B 1410 1FA$="X"THENSY3651OB
01 1420 GOTO1370
1F 1430 PRINT"{C4 CLR}" POKES32
   B8 15 POKE53281,15
93 1440 PRINTTAB15)"{DOWN2,CA S
   *20 ES}"
C2 1450 PRINTTAB15)"{S-,SPC3}S
   I O R A G E{SPC4,S-}"
41 1400 PRINTTAB15]"CO,5*20,CW
   }"
45 1470 PRINTTAB151"{S-1 STORA
   GE TO  "OB1DU)"  {S0}"
55 1480 PRINTTAB{S1"CCO,5*20,CW
   }"
AA 1490 PRINTTAB15]"{S-}CHANGE
   DEV1CE{SPC4}  {WH1E}DCC4 S0
   }"
CC 1800 PRINTTA0151"{S-}SAVE CH
   ARACTERS   {WHITE}CCC4Y S0]"
C4 1510 PRINTTAB15]"{S-}SAVE IE
   XI{SPC0}  {WH1E}IC{4,S0}"
40 1520 PRINTTA0151"{S-}SAVE MA
   CH1NECODE ; {WH1E}RKC4 S0}"
84 1530 PRINTTA015]"1S 3LOADC{SP
   C13}  {WH1E}LCC4 S0}"
90 1540 PRINTTA015]"CCQ 5*20 CW
   }"
90 1550 PRINTTA151"{S-}EX1TCSP
   C13}  {WH1E}XCC4,S-}"
98 1560 PRINTTA0{S1"CCZ 5*20 CX
   }"
5A 1570 GETA$ 1FA$=""THEN1570
E5 1580 1FA$="D"TKEN1630
7C 1590 1FA$="C"THEN1670
9F 1600 1FA$="1"THEN1600
E3 1610 1FA$="I"THEN1650
41 1620 1FA$="L"THEN1790
B2 1630 1FA$="X"THEN800T0120
54 1640 GOTO1570
EB 1650 1FDU=1THENDU=2 DE=8 PRI
   NT"{HOME,DOWN3}"TAB124)DS{DU
   } GOTO157B
64 1660 DU=1 DE=1 PRINT"{HOME,D
   OWN6}"TAB124)DS1DU} GOTO1570
98 1670 N8=" CHARACTERS" SA=815
   2 FA=16241 GOSUB1150 GOTO1430
AC 1000 N8=" MACHINE C " SA=451
   52 FA=49510:GOSUB1150 GOTO14
   30
41 1690 N8="{SPC7}TEXT" SA=2048
   FA=PEEX(45336)*256 GOSUB115
   0 GOTO1430
7B 1700 PRINT"{CLR,DOWN2}"IA011
   01"LOAD" PRINTTA013B1"{CT4}"
33 1710 1NPUT"{DOWN2 CS}O1VE F1
   LENAME",F1$ LOADF1$,DE,1 A8=
   "" GOTO1430
4C 1720 PRINT"{CL01" POKE53280,
   12 POKE53281 1S
B0 1730 PRINTTA015]"1C4,DOWN3,C
   A,5*20,CS}"
D7 1740 PRINTTA0{S]"{S-} CCSJT
   E X TCSPC4}M E N UCC4} {S-}"
52 1750 PRINTTA101"CCQ 5*20,CW
   }"
0B 1760 PRINTTA015]"{S-}ED1I IE
   XI{SPC0}  {WH1E}IC{4,B0}"
A1 1770 PRINTTAB{R}"{S-}VIEW IE
   XTCSPC0}  {WH1E}VC{4,B0}"
82 1780 PRINTTAB{S}"{S-}VIEW SC
   ROLL1NG{SPC3I  {WH1E}3CC4 B
   01"
4E 1790 PRINTTA0{S}"{S-}REPLACE
   TEXTCSPC5I  1WH1TE1R1C4 S0]
   "
41 1000 PRINTTA0191"{S-}SET MES
   S  LENGTH   {WH1TE}MCC4 S01"
E1 1010 PRINTTAB151"CCO 5*20 CW
   }"
09 1820 PRINTTAB151"{S-1EXIT{SP
   C131  {WHITE1XCC4,S-}"
01 1030 PRINTTAR{S1"{CZ_8*20 CX
   1"
5B 1840 DETAS 1FA$=""THEN1040
50 1050 1FA$="E"TKEN1920
92 1900 1FA$="V"THEN2110
2E 1870 1FA$="S THEN2300
56 1000 1FA$="M"THEN2400
70 1890 1FA$="R"TKEN31RR
7E 1000 1FA$="X"THEN1230
01 1910 GOTO1840
07 1920 PRINT"{CLR C4,DOWN2}ALT
   ER START ADDRESS ? 1Y/N1 DEF
   =2049" /R=PEEX(45336)
7F 1930 DETA8:1FA8=""THEN1B30
D0 1940 NA=2048 1FA$<>"Y"THEN1R
   70
46 1050 1NPUT"{CLR DOWN C4}GIVE
    NEW RTART ADDRESS";NA
1B 1960 1FNA<{8*8}ORNA>{19*256}TKE
   N1950
E0 1970 PRINT"{CLR DOWN C4}CKAR
   ACTERS FREE "{19*256-NA PRINT
   "{DOWN1EX1T {WH1TE}{RETURN}
   {C4}"
00 1305 2$="" Z=0:PRINT"{8DOWN2,
   CS}1NPUT TEXT CC41 -{LEFT}",
08 1990 GETA$ 1FA$=""THEN1990
D7 2000 1FAS=CHR$(113)THENGOTO17
   20
81 2010 1FAS=CHR$(28)ANDZ>0THEN
   PRINTAS"-{LEFT1"  Z=Z-1:GOTO
   1990
C7 2020 1FAB>CHR$(131)ANDA$<CHR$
   (16)1THENVA=ASC(A$) GOTO207B
D4 2030 1FAS>CHR$(163)ANDA$<CHR$
   (198)THENVA=ASC(A$)-64 GOTO20
   7B
12 2040 1FAS>CHR$(155)ANDA8<CHR
   $(222)1THENVA=ASC(A$)-128 GOT
   O207B
4C 2050 1FAS>CHR$(155)ANDA$<CHR
   $(152)THENVA=ASC(A$)-64 GOTO
   2078
BB 2060 GOTO1990
FF 2070 PRINTAS  POKENA+Z,VA Z=
   Z+1
D9 2080 1FZ+NA->A8*256THENPRINT
   "PRINT"CDOWN}MEMORY FULL GO
   TO2330
75 2090 1FAS=CHR$(34)1THENPRINTC
   HR$(2B)1CHR$(34),
E2 2100 PRINT"-{LEFT}" ;GOTO109
   0
A5 2110 1NPUT"{CLR,DOWN2 C4}DIV
   E START ADDRESS ? 20*0{LEFT6
   }";TI MS=PEEX(14336)1
73 2120 1FTI<20*60ORTI>MS*256THE
   N2110
8F 2130 PRINT"{DOWN2 C4}PRE
   SS {WH1TE}RETURN}{C4} TO RT
   ART"
1A 2140 PRINT"PRESS {WH1TE}{P7}
   {C4} TO PAUSE AND RESTART"
F6 2150 PRINT PRESR {WHITE}{SPA
   C2}{C4} TO EX1T" AD=2040
A4 2100 GETA8:1FA8<>CHR$(113)THE
   N2160
20 2170 PRINT"{DOWN} ;FORX=TITO
   MS*256RTEP24
7E 2100 X8=STR$(X+1:X8=R1GHT$(X8
   ,LEN(X$)-1) PRINTTAB(5-LENCX
   $)11"{BLACK}"X$"{SPC5 C4}";
0B 2190 FORY=0TO23
FF 2200 B=PEEX(X+Y):1FD>31ANDD<
   64THEN225B
AB 2210 1FD=>0ANDD<3ETHENQ=D+64
```

```
    GOTO2250
07  2220 IFQ=9ANDQ<120THENQ=Q+
    64:GOTO2250
71  2230 IFQ=64ANDQ<86THENQ=Q+3
    R:GOTO2850
7A  2240 PRINT"{CS SU_C4}";:GOTO
    2250
0R  R250 PRINTCHR$(Q);:IFQ=34THE
    NPRINTCHR$(20);CHRR(34);
30  2260 GETA$:IFA$<>""THEN2200
F0  2270 NEXT PRINT NEXT GOTO223
    0
7F  2280 !FA$="{F7}"THEN2310
03  2290 !FAR=" "THENE330
F4  2300 GOTO2270
DE  R310 GETAR !FAR<>"{F7}"THENR
    310
I7  23R0 GOTO2270
FF  2330 PRINT:PRINT"{DOWN2,SPC9
    }PRESS {WHITE}{SPACE}{C4} T
    O EXIT"
0B  R340 GETA$:IFA$<>" "THENR340
3A  R350 GOTO1720
3A  2360 PRINT"{C4 CLR}" POKE532
    80,15 POKE53281 15:POKE889,0
    :POKE900 189 POKE901,0
AB  R370 POKE49324,0 SYS49152:PR
    INT"{HOME DOWN4 C4,SPC9}PRES
    S {WHITE}{SPACE}{C4} TO EXIT
    "
27  2380 GETA$ !FA$=" "THEN2310
43  2390 GOTO2380
F1  2400 PRINT"{CLR}"  MS=PEEK(4
    9286)
35  2410 PRINT"{HOME DOWN2,C4}ME
    SSAGE LENGTH (WHITE)"25B*(MS
    -81"{LEFT,C4} CHARACTERS "
E0  2420 PRINT"{DOWN2}PRESS {WHI
    TE}+{C4} TO INCREASE AND {WH
    ITE}-{C4} TO DECREASE"
2C  2430 PRINT"NUMBER OF CHARACT
    ERS"
03  2440 PRINT"{DOWN2}PRESS {WHIT
    E}{SPACE}{C4} TO EXIT"
BD  2450 GETA$ IFA$=""THEN2450
D2  2460 IFAR="+"ANDMS<32THENMS=
    MS+1 POKE49386 MS GOTO2410
55  2470 IFAR="-"ANDMS>8THENMS=M
    S-1 POKE49385 MS GOTO2410
1F  2480 IFA$=" "THENGOTO1720
4D  2490 GOTO2450
03  2500 PRINT"{C4,CLR}" POKE532
    84,0 POKE53277,0 POKE53271,0
     R=0 S=0 T=0 TT=24 MM=1
0C  2510 MD=224 FORX=8TO4:POKE10
    48+X CH=X POKE1080+X,CH+X+12
    0 NEXT A1=CH A2=A1+4
EE  R520 PRINT"{HOME}"; FORX=1TO
    16 PRINT"{C4 SO40}";
B0  2530 NEXT
C3  2540 PRINT"{RVSOFF C4} FM {W
    HITE}""M""{C4 SPC3}POKE{WHI
    TE}*A1"{C4}TO{WHITE}"A2"{C4
    SPC3}EXIT {WHITEX}"
AE  2550 P=PEEK(56320)1 IF(PAND1
    1=0ANDS>0THENS=S-1
0C  2560 IF(PAND2)=0ANDB<15THENB
    =B+1
AD  2570 IF(PAND4)=0ANDT=0ANDR=1
    THENT=R0:R=0 TT=24 POKE53204
     0 GOTO2620
10  2580 IF(PAND4)=0ANDT>0IXENT=
    T-1
07  2590 IF(PAND8)=0ANDT=20THENB
    =1 TT=0:POKE53204 1:T=0 GOTO
    2620
AC  2600 IF(PAND8)=0ANDR=0THENT=
    T+1
0A  2610 IF(PAND8)=0ANDR=1ANDT<1
    0THENT=T+1
7E  2620 IF(PAND16)=0THENPOKE128
    4+S*40+T+2D*R+R*1,MD
3E  2630 POKE53240,TT+T*8 POKE53
    249,50+5*8 GETA$ IFA$=""THEN
    2550
5B  2540 !FA$="C"THENE520
51  2650 !FA$="+"THENMD=224 MM=1
    PRINT"{UP}", GOTO2540
7C  2660 !FA$="-"THENMD=79 MM=0
    PRINT"{UP}", GOTO2540
7B  2670 !PA$="X"THENRESTORE GOT
    O20
20  2680 !PA$=" "THEN2710
2B  2650 !FA$="{P}"THENBOTO3020
A0  2700 GOTO2550
25  R710 POKES3200,0
47  2720 FORW=0TO4 FORY=0TO7 POR
    X=7TOBSTEP-1 Z=PEEK(1024+W*8
    +(7-X1+Y*40)
01  2730 ZZ=8192+CH*8+W*8+Y 1FZ-
    224THENPOKE22,PEEK(ZZ)OR(2^X
    1 GOTO2750
30  R740 POKCZ2,PEEK(ZZ)AND(255-
    2 X)
C2  2750 NEXT NEXT NEXT
0A  2760 FORW=0TO4 FORY=0TO7 FOR
    X=7TOBSTEP-1 Z=PEEK(1344+W*8
    +(7-X1+Y*40)
04  2770 ZZ=8192+CH*8+W*8+Y+128*
    8 IFZ-224THENPOKE22,PEEK(ZZ)
    OR(2^X):GOTO2750
91  2780 POKE22 PEEK(ZZ)AND(255-
    2^X)
7B  2790 NEXT NEXT NEXT POKES320
    0,11 BOTO2550
E0  2600 PRINT"{HOME}" FORK=1TO1
    5 PRINT NEXT PRINT"{BLACK}15
    CHARS FROM" WO-1 BOSUB450 WQ
    =0
17  2810 CH=QQ BOTO2500
02  2820 PRINT"{HOME}" FORK=1TO1
    5 PRINT NEXT PRINT"{BLACK}MI
    RROR WHICH CHARACTER" WO-1
B0  2830 BOSUB450 POKES3200,0
01  2840 QQ=B+8192 WQ=0 POKC53
    265,0 POKES3248,24+X*8 POKES
    3245,50+Y*8
AC  2850 PRINT"{HOME,BLACK}" FOR
    X=1TO15 P0INT NEXT POINT"{UP
    ,SPC24}"
32  2860 FORU=0TO7 POKE350+Q,PEE
    KC2+01 POKES60+Q,PEEK(2+128*
    8+Q1 NEXT
97  2870 FORK=0TO7 QQ=PEEK(350+K
    1:ZZ=Z+K FORL=0TO7
25  2880 !FQQ-(2 (7-L))=>0THENPO
    KE22 PEEK(221OR(2^L) QQ=QQ-(
    2 (7-L1) BOTO2560
56  2890 POKE22,PCEK(ZZ)AND(255-
    2 L)
2B  2900 NEXT NEXT
B9  2910 FORK=0TO7 QQ=PEEK(500+K
    ):ZZ=Z+128*8+X:FORL=0TO7
75  2920 !FQQ-(R (7-L11=>0THENPO
    KE22,PEEK(2210R(2 L) QQ=QQ-(
    2 (7-L1) BOTO2940
0E  2930 POKE22,PEEK(221AND(255-
    2^L)
B7  2940 NEXT:NEXT POKE53200,11
    BOTO90
4E  2950 POKE53200,0
1E  2960 PRINT"{HOME}" FORK=1TO1
    5 PRINT NEXT PRINT"{BLACK}MI
    RROR WHICH CHARACTER" WQ=1
FF  2970 BOSUB450 Z=QQ-8+8192:WQ
    =0 POKES3264,0 POKE53248,24+
    X*8 POKE53245,50+Y*8
22  2980 PRINT"{HOME,BLACK}" FOR
    K=1TO16 PRINT NEXT:PRINT"{UP
    ,SPC24}"
14  2990 FORQ=0TO7 POKE950+Q,PEE
    K(Z+Q1 POKES60+Q,PEEK(Z+8+12
    8+01 NEXT
29  3000 FORQ-7TOBSTEP-1 POKEZ+Q
    ,PEEK(957+Q1 POKEZ+8+128*8,P
    EEK(957-Q1 NEXT
55  3010 POKE53200,11 BOTO60
E4  3020 PRINT"{HOME}", FORX=1TO
    16 PRINT"{C4 SO40}",
63  3030 NEXT
C0  3040 POKE53200,0
0A  3050 FORW=0TO4 FORY=0TO7 OO=
    8192+CK*8+W*8+Y Z=PECK(QQ1 Z
    Z=PEEK(CQ0+128*81
67  3060 FORX=0TO7
BC  3070 IFZ-R (7-X)=>0THEN2=2-2
    ^(7-X1 POKE1024+W*8+Y*40+X R
    E4
2B  3080 IFZZ-R^(7-X1=>0THEN2Z=2
    2-2 (7-X1 POKE1344+W*8+Y*40+
    X,224
D3  3090 NEXT:NEXT NEXT PRINT:PO
    KES3200,11:BOTOR550
B5  3100 PRINT"{CLR,DOWN2,C4}"TA
    B(14)"REPLACE TEXT":PRINTTAB
    (14)"{CI2}"
E4  3110 INPUT"{DOWNR,C4}DIVE TE
    XT START ADDRESS",TS
77  3120 INPUT {DOWN}DIVE  TEXT
     END ADDRESS" TE
1B  3130 INPUT"{DOWN}DIVE NEW  S
    TART ADDRESS",NS
C3  3140 IFTE<TSTHENPRINT"{DOWN,
    WHITE}END SHOULD BE HIGHER T
    HAN START" BOTO3240
2A  3150 IFTS<2048BORTS>0151THENG
    OTO3230
9C  3160 IFTE<2048BORTE>0151IHENG
    OTO3230
0A  3170 IFNS<2048BORNS>0151THENG
    OTO3230
A8  3180 IFNS+TE TS>0151THENPRIN
    T"{DOWN,WHITE}NEW START ADDR
    ESS TOO HIGH" BOSUB3240 GOTO
    3180
46  3190 IFTE-TS>2000THENPRINT"{
    DOWN,WHITE}TEXT TOO LONG (LE
    NGTH <= 2000)" GOTO3240
58  3200 PRINT"{DOWN,BLACK}PLEAS
    E WAIT" POKES3200,0
EA  3210 K=TE TS FORL=0TOK POKE1
    024+L PEEK(TS+L1 NEXT FORL=
    0TOK
0E  3220 POKEN54 PEEK(1024+L1)
    NEXT POKES3200,11 GOTO1720
BC  3070 !F2-2 (7-X)=>0THENZ-2-2
    ^(7-X1 POKE1024+W*8+Y*40+X-2
    24
2B  3080 !F22-2 (7-X)=>0THENZZ-2
    2-2 (7-X1 POKE1344+W*8+Y*40+
    X,224
03  3050 NEXT NEXT NEXT PRINT:PO
    KES3200,11:GOTO2550
B5  3100 PRINT"{CLR,DOWN2,C4}"TA
    B(14)"REPLACE TEXT":PRINTTAB
    (14)"{CI2}"
E4  3110 INPUT"{DOWN2,C4}DIVE TE
    XT START ADDRESS",TS
77  3120 INPUT"{DOWN}DIVE  TEXT
     END ADDRESS",TE
1B  3130 INPUT"{DOWN}DIVE NEW  S
    TART ADDRESS",NS
C3  3140 IFTE<TSTHENPRINT"{DOWN,
    WHITE}END SHOULD BE HIGHER T
    HAN START" BOTO3240
2A  3150 IFTS<2048BORTS>0151THENG
    OTO3230
9C  3160 IFTE<2048BORTE>0151THENG
    OTO3230
0A  3170 IFNS<2048BORNS>0151THENG
    OTO3230
A8  3180 IFNS+TE-TS>0151THENPRIN
    T"{DOWN,WHITE}NEW START ADDR
    ESS TOO HIGH" BOSUB3240:BOTO
    3180
46  3190 IRTE-TS>2000THENPRINT"{
    DOWN WHITE}TEXT TOO LONG (LE
    NGTH <= 2000)" OOTO3240
```

```
50  3200 PRINT"{DOWN,BLACK}PLEAS
    E WAIT":POKE53280,0
EA  3210 K=TE-TS:FORL=0TOK POKE1
    024+L,PEEK(TS+L) NEXT FORL=
    0TOK
DE  3220 POKENS+L,PEEK(18240+L)
    NEXT POKE53280,11 GOTO1720
B6  3230 PRINT"{WHITE,DOWN}TEXT
    MUST BE BETWEEN 2048 AND 819
    2 !"
B5  3240 FORL=1TO2000:NEXT:GOTO3
    100
```

PROGRAM: M/C MAKER

```
24  10 BL=33      LN=50      SA=4519
    2
C0  20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
67  30 POKE53280,A:POKE SA+L*16+
    D,A NEXT D
73  40 READ A IF A>=CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
C4  45 NEXT L SYS45512
3B  50 DATA 120,169,31,141,13,22
    0,141,13,221,173,13,220,173,
    13,221,173 2855
13  60 DATA 17 206 41 127 141 17
     208 169,150,141,20 3,169,19
    2,141,21 1813
B4  70 DATA 3,169,1,141,26 200,1
    65 56,141,10 200,96,96,165 2
    00,141 1044
02  80 DATA 22,200 169,211,141,2
    0 3 169 182,141 21,3,165 251
    ,141 19 1078
03  90 DATA 200 169 1,141 25 200
    ,76 45,234 169 199,141,22 20
    0 169 199 2217
EE  100 DATA 141,20 3,169 182 14
    1 21,3 165 86,141 18 200 169
    ,1 141 1603
F6  110 DATA 25 200 234,234,234,
    174,131,3,232,224,1,240,6,14
    2 131 3,2222
0E  120 DATA 76 45,234,162,0,142
     131,3,174,132,3 202 224,191
    200 3,1934
C4  130 DATA 76 140 182,142,22,2
    96 142 132 3 76 49 234 162,1
    99 142 132 2051
01  140 DATA 3 169 0 185 1,4,153
     0,4 185 41,4,153 40,4,200,1
    137
G0  150 DATA 192 40,240,3,76,147
     192,174,133,3,169,0,0,141,3
    3 4 1581
D0  160 DATA 185 127,141,79,4,23
    4 234 234 234,234,232,142,13
    3,3 165 133 2584
FE  170 DATA 141,74 162,76,218,1
    92,150,2,136,208,253,169,21,
    141,24 200 2210
17  180 DATA 76 45 192,169,26,14
    1,24,200 76 73 192 174 133,3
    ,224,0 1755
B7  190 DATA 240,3,76 45,234,174
    ,172,152,232,224,12,208 2,16
    2,0,142 2132
B4  200 DATA 172,152,76,45,234 1
    28,169,31,141,13 220,141 13
    221 173 13 1976
00  210 DATA 220 173 13,221 173
    17 208 41 127,141,17 208 165
    ,34 141 20 1523
E0  220 DATA 3,169 153 141 21,3,
    169 1 141 25 200 169 50,141,
    10 200,1601
70  230 DATA 80 96 169 21,141 24
    ,200 169 82 141 20,3,165,193
    141,21 1690
26  240 DATA 3,169,186,141 10 20
    0,169 1,141 25,200 76,49,234
    ,234,234 2096
B0  250 DATA 234,234,234,234,234
    ,234 234 169,25,141,24 200,2
    30 39 200 169 2058
9C  260 DATA 34 141,20 3,169 193
    ,141,21,3,169,50,141,18,200,
    169,1 1401
52  270 DATA 141,25,200,76,49 23
    4,0,0,165,0 133,250,169,192,
    133 251,2030
0C  280 DATA 169,0,133,174,133,1
    33 165,152,133,175,133 174 1
    69 184,133,252 2156
F5  290 DATA 169 153 133 253,166
    ,0 177,200,145 174,252 200,2
    50,2,238,251 2025
0A  300 DATA 238 174,200,2 230,1
    75,165 250,197 252 200 234,1
    85 251,197 253,3191
49  310 DATA 208 229,165 3,133,1
    07,165,134 133,150 165,3,133
    ,163,165 0,2269
AB  320 DATA 133,165,150,0,165,2
    11 133 248 5 32 210,255,200,
    200,245-32 2435
20  330 DATA 207 255 248 251 201
    49 240 4 201,58 48,230 41,1
    5 133,150 2357
A0  340 DATA 76,234 245,147,17,1
    7,73 70,58,85,84,32,98,83,98
    ,73,1464
30  350 DATA 67,69,32,78 85 77,6
    5 69 32 13 17 67,65,83 61,48
    ,500
70  360 DATA 32,47,32 98,73,83 7
    5 61 32,98 32 78,82 32 57 58
    ,855
90  370 DATA 45,32,0,77 47,87,0,
    0,0,0,0 0,0,0,0 208
AD  380 DATA 0 0,0,255 255,255 2
    55,0,0,0,0,0,0,0 0,0,1020
```

PROGRAM: CHARACTER MAKER

```
25  10 BL=139     LN=50      SA=3686
    4
C0  20 FOR L=0 TO BL CX=0:FOR D=
    0 TO 15 READ A CX=CX+A
B7  30 POKE53280,A POKE SA+L*16+
    D,A:NEXT D
73  40 READ A IF A>=CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
C5  45 NEXT L SYS36520
72  50 DATA 60,102,66,126 106,17
    0,102,102,124,254 198 198,26
    4,254 198,102,2640
90  60 DATA 252,254 198,198,198,
    252 252 192 48 124 110,198,1
    98,192 192,3584
0A  70 DATA 240 200,12 54,94 54,
    54,102,60 126,102,102 102 22
    4,112 50,1934
6D  80 DATA 252,254 198 198 192,
    192 198,60 28 52 110 102 60 1
    52,192,192,2450
B5  90 DATA 192 192,192,192,60,0
    0 60 124,24,24,0,24,24,24,24
    24 1348
D0  100 DATA 12,12 0,12 12 12 12
    6,192 192 192,192 192 204,2
    20,248 1710
61  110 DATA 192,192,96,60 60,48
    ,48 48 192 198 254,254,214,1
    98,198,198,2460
90  120 DATA 228,204,230,198,198
    198 192 192,60,120 192 192,
    198,198,198,192 2604
3A  130 DATA 248,252 208 198 198
    ,102,118 124,56 126 192 192,
    198 198 198 198,2612
D1  140 DATA 240,252,12,198,198,
    198 204 252,56 124 230,198,1
    92 224 120 28 2742
44  150 DATA 56 56 240,120,56,56
    ,56 152,198 198 198 198,198,
    198 198,198 2616
C9  160 DATA 198 198 198 198,198
    ,198,198,198,12,14,0,192,192
    ,198,198 198 2144
55  170 DATA 198,198,198,198,198,
    198,198 56 198 198 198 198,
    198 192 126,30 2610
16  180 DATA 254 254,6 6,14 12 2
    0,56,96,96 96 96 96,48 48 48
    ,1830
06  190 DATA 120 252 238 192,192
    ,192 60,104 60 60 12,12 12 1
    2,12,12 1884
03  200 DATA 16,16,56,56,124 124
    ,204 254 16,16,16 48 48 112
    112 204 1582
3D  210 DATA 0 0 0 0 0 0 0 0,60
    60 60 60 60 60,24,444
CF  220 DATA 230 230,34,102 254
    0,0 0 54,54 54,54,54,236 252
    120 1750
54  230 DATA 60 124 204,222,216
    248,120 124,96,96,120,102,1
    2 12 24,24 1536
73  240 DATA 48 120 72 216 200 1
    44 240,96 56,56 0 24 48,0 0,
    0,1336
BE  250 DATA 28,56,96 96,192 192
    192,192 112,56 12 12,6,6 6,
    6,1260
F8  260 DATA 0 0,0,0 0,16,84,60,
    0,0 0,24 24,24 126,354
A6  270 DATA 0 0,0,0 0,0,0 0 0,0
    0,0 0 0 254,254
F8  280 DATA 0 0 0 0,0 0 0 0,2,2
    ,6 6,12 12 24 24 98
C9  290 DATA 124 198 198 198 198
    198,130,0 0 2 6,6 6 6 2 0,1
    250
6E  300 DATA 124 98 6 6,6 6,60,1
    24,124,60,6 6 6 6,60 124 776
B7  310 DATA 0,130,158,158,198 1
    98,198 124 124,124,182 182,1
    92,182,184 124,2616
B0  320 DATA 124 184 182 192 192
    ,182,184 124 124,60,6,6 6 6
    2,0 1592
0C  330 DATA 124,198,198 192,198
    ,198,186 124 124 186 198 198
    ,198,198 124 2824
C0  340 DATA 0 0 0 56 56 56 0 0
    0,0 0 56 56,0 0 336
AC  350 DATA 0,0 0 0 0 2,12,48,192
    ,0 0,0 0 0 124,124,0,992
6F  360 DATA 0 0 0 0 0 120 36 24,6
    ,124 254 198 198 182 6 12 24
    ,1176
F8  370 DATA 0 0,0 0 0,1 2,63,64 0
    ,0 0 255 37 65,200 5 746
C4  380 DATA 0 0 0 0,255 5,4 2 255
    ,0 0 0 224,24,132 130 128,11
    56
74  390 DATA 0,0,0,0,0,0,0,0,224,2
    24 24 4 2 2 1 1 1 483
C0  400 DATA 0,0 0 0 0 0 0 0,0,0 0
    ,0,0 0,1 1 1,3
78  410 DATA 15,40 64,128 128,0,0
    0,0,15 63,112 90 192 192,192
    ,192 1437
BD  420 DATA 128 128,128,124-128
    ,112,96 0 205,127,191,95,179
    ,87 171-85,2828
DF  430 DATA 255 254,255 252,248
    ,226,196 144,137 82,34 34 33
    ,0 0 0 2149
1A  440 DATA 146,82,82 82 142,0
    0,0,224,144,225 161,145 3 3,
```

```
3A 3,1442
   450 DATA 232,249,145,147,18,
 6 8,9,159,227,35,51,19,27,27
 ,28,1420
E8 460 DATA 31,25,25,153,183,16
 3,163,163,142,140,140,204,21
 2,213,213,85,2259
B6 470 DATA 48,120,72,204,132,1
 34,134,134,240,240,144,152,1
 36,141,141,141,2314
08 480 DATA 48,120,72,204,132,1
 34,134,134,241,240,140,140,1
 32,140,140,240,2400
03 490 DATA 252,132,128,128,128
 ,128,136,248,248,252,252,132
 ,184,148,188,132,2750
9A 500 DATA 7,24,32,70,76,140,1
 38,130,224,24,4,82,90,97,1,6
 5,1172
3D 510 DATA 1,3,8,15,21,32,190,
 120,255,255,170,256,85,0,170
 ,0,1582
36 520 DATA 128,192,160,240,80,
 4,170,1,0,3,25,62,90,67,12,1
 2,1250
49 530 DATA 0,0,112,240,136,250
 ,224,90,0,0,0,1,1,183,192,64
 ,1467
92 540 DATA 31,190,177,49,255,3
 5,4,4,0,192,160,144,240,144,
 0,0,1542
DD 550 DATA 31,17,17,245,128,18
 ,1,128,245,0,0,0,224,32,156,3
 2,224,1984
D4 560 DATA 63,84,90,156,156,14
 4,140,120,240,4,52,114,210,1
 0,90,2,1879
D3 570 DATA 1,1,0,0,0,0,127,82,
 0,152,241,30,90,255,128,65,1
 207
EE 580 DATA 0,0,0,128,144,240,206
 ,132,124,7,4,9,12,63,187,231
 ,155,1986
90 590 DATA 128,0,0,54,128,192,22
 5,89,48,30,33,73,112,28,210
 ,90,130,1565
15 600 DATA 3,2,4,132,72,49,2,6
 ,0,136,38,68,130,140,70,76,1
 ,803
C0 610 DATA 0,0,0,0,0,0,0,0,0,0
 ,0,12,20,60,150,254,412
D4 620 DATA 0,0,0,48,24,28,23,1
 ,0,0,0,0,103,154,104,66,66,76
 ,6
D6 630 DATA 0,0,0,16,27,33,32,6
 4,255,131,131,131,255,254,25
 3,253,1834
C0 640 DATA 255,255,255,255,050
 ,63,223,223,255,255,192,128,
 128,143,146,146,3177
00 650 DATA 255,255,0,0,0,255,2
 25,225,255,255,3,1,1,241,73,
 73,2117
F0 660 DATA 255,112,32,32,32,33,33
 ,33,164,255,0,125,115,10,2,2
 ,130,1353
7D 670 DATA 252,0,207,190,30,30
 ,36,30,255,0,223,210,02,2,2,
 2,1430
74 900 DATA 255,7,133,153,85,0,
 0,0,131,60,60,90,00,00,56,50
 ,1322
1D 690 DATA 102,162,170,190,145
 ,10,100,56,180,102,0,6,12,12
 ,12,24,1334
0E 700 DATA 102,102,102,102,190
 ,12,124,120,192,102,190,190,
 190,204,252,120,2326
3B 710 DATA 102,102,190,190,190
 ,190,254,252,192,192,102,102
 ,150,190,254,124,3044
FE 720 DATA 120,240,90,40,40,40
 ,40,40,192,000,222,190,190,2
 00,252,120,2250
53 730 DATA 126,110,102,102,102
 ,102,102,102,24,24,40,40,40,
 40,112,56,1304
E0 740 DATA 6,6,6,190,190,102,1
 00,60,240,240,120,100,100,10
 0,110,102,1046
12 750 DATA 40,40,40,56,56,90,2
 54,252,102,6,6,6,12,12,12,24
 ,1120
3E 760 DATA 102,6,6,6,12,12,20,
 24,190,190,190,190,190,102,1
 24,56,1480
AF 770 DATA 120,96,40,40,40,40,
 40,40,190,190,214,222,222,66
 ,0,254,114,2140
BC 780 DATA 125,190,190,102,102
 ,190,190,190,12,14,6,0,190,2
 30,120,56,1750
90 790 DATA 192,152,152,190,190
 ,110,124,56,190,190,190,190
 ,190,230,125,62,2670
EA 800 DATA 190,190,56,56,92,56
 ,16,10,190,190,190,190,214,2
 22,246,152,2840
70 810 DATA 190,190,190,190,190
 ,190,190,190,0,6,6,6,12,12,1
 2,24,1490
BF 820 DATA 112,90,224,102,102,
 192,254,254,40,40,40,40,40,4
 0,90,58,1924
C1 830 DATA 240,240,90,90,90,12
 2,254,204,12,12,12,12,12,12,
 60,56,1540
DD 840 DATA 56,56,56,90,56,56,5
 6,56,254,112,112,40,40,40,10
 ,16,1192
AF 850 DATA 0,0,0,0,0,0,0,0,24,
 24,24,24,0,24,24,168
16 860 DATA 0,0,0,0,0,0,0,0,238
 ,252,126,206,216,216,216,216
 ,1886
C2 870 DATA 60,62,54,54,246,250
 ,252,240,40,40,90,64,204,210
 ,146,140,2178
2D 880 DATA 224,162,192,192,148,150
 ,156,222,114,0,0,0,0,0,0,0,0
 ,1364
DB 890 DATA 132,102,150,150,90,
 90,56,20,6,6,6,6,12,12,56,11
 2,1206
FC 900 DATA 254,56,84,10,0,0,0
 ,0,126,24,24,24,0,0,0,0,1005
49 910 DATA 0,0,0,56,56,0,24,40
 ,254,0,0,0,0,0,0,0,445
90 920 DATA 0,0,0,0,0,56,56,56,
 40,40,90,90,102,192,125,120,
 1050
E9 930 DATA 130,150,150,150,150
 ,190,190,124,2,6,6,6,6,2,0
 ,1454
B4 940 DATA 104,192,190,192,102
 ,102,104,124,50,6,0,0,0,0,02
 ,124,1722
1F 800 DATA 56,6,6,6,6,6,2,0,92
 ,6,6,6,6,92,124,360
97 960 DATA 150,150,150,150,150
 ,190,190,124,2,6,6,0,0,0,2,0
 ,1520
63 970 DATA 150,150,190,190,190
 ,190,106,124,50,0,0,0,6,0,50
 ,124,1750
83 900 DATA 0,0,0,56,56,92,0,0,0
 ,0,92,50,56,0,24,40,410
EA 990 DATA 152,40,12,2,0,0,0,0
 ,0,124,124,0,0,0,0,0,502
9E 1000 DATA 6,24,90,120,0,0,0,
 0,24,24,24,24,24,0,24,24,422
26 1010 DATA 134,135,125,225,25
 5,30,0,0,244,10,243,141,110,
 213,210,112,2217
32 1020 DATA 2,20,2,2,256,255,0
 ,0,4,195,15,24,247,237,13,7,
 1285
5F 1030 DATA 20,2,123,193,126,1
 00,120,0,1,1,1,0,0,0,0,0,0,803
AE 1040 DATA 0,0,0,120,120,64,4
 0,15,1,1,1,2,2,4,24,224,542
CF 1050 DATA 0,0,0,0,0,0,0,0,10
 2,192,192,192,96,112,63,15,1
 054
41 1060 DATA 0,32,80,40,84,42,1
 28,120,170,84,160,80,160,64,
 120,0,1300
48 1070 DATA 65,4,16,1,4,0,1,0,
 0,0,0,0,0,0,0,0,31
D8 1080 DATA 0,0,0,0,0,0,0,0,3,
 3,1,1,1,0,0,12
FD 1550 DATA 6,0,0,16,11,145,24
 1,224,25,26,26,10,02,34,226,
 193,1254
0F 1100 DATA 103,155,201,251,20
 1,201,73,90,05,06,101,150,10
 0,150,36,174,2901
47 1110 DATA 134,134,134,132,25
 4,72,120,48,141,141,141,130,
 192,144,240,240,2314
F6 1120 DATA 134,134,134,132,25
 4,72,120,48,170,176,162,150,
 150,152,140,207,2266
25 1130 DATA 240,156,120,120,12
 0,120,132,252,252,132,100,13
 2,252,225,262,249,2556
8F 1140 DATA 153,176,172,67,64
 ,32,24,7,145,3,57,194,2,4,24,
 224,1354
90 1150 DATA 120,106,38,21,15,6
 ,3,1,0,170,0,25,255,170,255,
 255,1592
98 1160 DATA 1,170,4,80,240,156
 ,152,120,24,8,16,24,24,0,24,
 24,1127
98 1170 DATA 65,130,12,0,1,2,1,
 3,224,248,208,208,84,32,16,4
 ,0,1254
9A 1190 DATA 4,4,4,4,4,4,4,31,0
 ,0,0,0,56,252,62,28,483
EC 1130 DATA 17,17,17,17,21,85,
 255,255,0,0,0,0,0,96,192,240
 ,1212
B7 1206 DATA 130,120,136,71,56
 ,36,16,15,130,2,34,196,90,72
 ,16,224,1343
69 1210 DATA 83,140,139,143,136
 ,120,65,62,34,147,253,255,25
 3,125,0,0,1070
16 1220 DATA 226,33,33,40,25,1,
 130,124,232,111,13,25,56,112
 ,97,49,1320
03 1230 DATA 105,166,230,136,19
 3,192,120,224,2,2,2,1,125,19
 2,0,0,1007
DF 1240 DATA 4,1,0,0,146,243,17
 ,68,82,4,132,148,24,240,0,0,
 1072
1A 1050 DATA 1,3,0,12,31,40,90,
 248,140,12,12,12,252,12,12,3
 0,910
B8 1260 DATA 32,32,32,32,65,65,
 55,227,132,132,132,132,0,0,0
 ,156,1250
92 1270 DATA 64,64,120,120,143,
 190,102,90,253,254,255,255,2
 55,255,255,2524
E7 1200 DATA 223,55,255,127,127
 ,127,127,255,143,120,255,255
 ,240,240,224,255,3044
FF 1250 DATA 255,0,255,255,0,0,
 0,255,241,1,255,255,31,15,7,
 255,2000
55 1300 DATA 252,164,32,32,32,3
 3,113,243,90,16,10,10,10,10,
 139,113,1315
10 1310 DATA 37,47,37,36,36,30
```

```
    100,220,2,2,2,2,2,2,7 556
FD  1320 DATA 0,0,0,0 117,39,37,
    37 60 60,69,69,20 60 20 60,7
    82
D7  1330 DATA 32,0,0,0,0,0,0,0,1
    60,0,133,250,169,144,133,201
    ,1201
78  1340 DATA 169,0 133,174,133,
    183 169 32 133,175,133,184,1
    69,0 133,252 2220
9D  1350 DATA 169 152 133 253,16
    0 0 177 250 145,174,250,250,
    200 2 230 251 2784
A2  1360 DATA 232,174,200,2,230,
    175,165 200,197,252,200 234,
    165 251 197,253 3191
43  1370 DATA 220,228,169 163,13
    3,107 169 152 133 120 169 5,
    133 183,169 0 2389
F5  1380 DATA 133,185 120,0,185,
    115 152,240,6 32 210 255,200
    ,208,245 32 2356
60  1390 DATA 207 255,240,251,20
    1 48,240 4,201 60,48,230 41,
    15 133 185 2357
00  1400 DATA 76,234,245,147 17,
    17 73 70 00,85,54,32,00,69,0
    0 73 1464
20  1410 DATA 67 69 32 76 85 77,
    66 69 82,13,17,67,65,03-61,4
    9 560
40  1420 DATA 32 47 32 60 73,83,
    75 61 32 55 32 70 82 32 57,5
    0 939
AE  1430 DATA 45 32,0,67,72,65,0
    2 83,0 0,0,0,0 0,0,0 446
B5  1440 DATA 0,0,0,255,255,255,
    255,0 0 0 0,0,0,0,0 0,1020
```

```
PROGRAM: TEXT MAKER

7D  10 BL=75      LN=50      SA=3614
    4
C8  20 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15:READ A CX=CX+A
B7  30 POKES3200 A POKE SA+L*16+
    D A NEXT D
73  40 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10) ST
    OP
CD  45 NEXT L 8Y536176
10  50 DATA 20 0 5 32,13,5,19,10
    ,1,7 5 32 3,15 14,19 217
5E  60 DATA 20 10 21 3 20 5 15,1
    4,32-11-9,20,32,23,1 15 267
E1  70 DATA 32 23 10,5 20,20,5,1
    4 32 2 25,32,8 10 1,14 271
E5  80 DATA 11 32 22 1 14 32 20,
    8,7,7 5 12 5 14 32 9 232
D3  90 DATA 14 32 13 1 18 3 8,32
    45 57 56 56 32 6,15 10 410
35  100 DATA 32 20 8 5,32,19,5,1
    8 5 15,21,10 32 21,19 5,200
14  110 DATA 10,39,15,32,7,21 9,
    4,5,40 40 48,13,3 11 32,351
D0  120 DATA 1,12-12,15,23,19,32
    ,1 14 25,15,14,5,32,20,15 25
    5
FD  130 DATA 32,3 10 5 1,20,5,32
    20 8 5 9 10 32 15,23,246
4C  140 DATA 14,32,10 13 15 15 2
    0,0,45,19,3 10,15,12,12 9 26
    9
66  100 DATA 14,7,44,32,9 14,20
    5 14 10 21,10,20,32 4,10,252

15  100 DATA 9,22,5,14,32,13,0,1
    9 15,1,7,5 19 32 23,0 233
D3  170 DATA 9 3,0,32,23,9 12 12
    ,32,5,22 5 14,32 10 21,207

2F  180 DATA 14 32,1,12,15,14,7,
    19 9,4,5 32 1,14,15,20 214
B7  190 DATA 0 5 10,32 15 10 15,
    7,18,1,13,46,46,46,46,46,301

B5  200 DATA 48,46,46,48,48,46,3
    2 20,0 5,10,5 32,13,5,10,434

20  210 DATA 15,1,7,5,19 32 3,15
    ,14 20,1,9,14 32,21,19 231
BA  220 DATA 5 10 32,4,5,0,9,14,
    5,4 32,7,10,1,16,0,124
00  230 DATA 9 3 15 32,1,14,4 32
    ,3 0 1 10 1 3,20,5,173
57  240 DATA 10,19,44,23,0,9,3,0
    ,32,1 10,5,32,5,1,10,245
BD  250 DATA 9 12 25,32 4,5 19,5
    ,7,14,5,4 32,21,19 5,220
23  260 DATA 14 7,32,20,0 5,32,9
    ,0,1,10 1,3,20 5,10 195
00  270 DATA 32 5 4 9 20,15 10 3
    2,32,32,32,32 32,32,32,32 32
    1
E5  280 DATA 32,32,32,32,32 20,5
    ,24 20,15,32 3,1,14,32,2 332
00  290 DATA 5,32,23,16,9,20,20,
    5 14 32,21,15,9,14,7,32,200
63  300 DATA 20,0,5 32 19,5,13,1
    6 12,5 32,20,5 24,20 32,272
59  310 DATA 5,4,9,20,15,10 32 4
    6 46,46,46,46,46,46,46 46 51
    7
58  320 DATA 32,1,32,6,5,23,32,5
    ,24,1,13,16 12,5,19,32,256
00  330 DATA 15,6,32 23,8,1,20,3
    2,9 19,32,16,15 19 9 275
C2  340 DATA 2,12,5,32,23,5 20,0
    ,32,13,3,11 32 50 32 32 324
C9  350 DATA 84,65,66,67,80,32-3
    ,8 9 12,4,9,19,8,32,16,482
D3  360 DATA 9 3,20,21,18 5,19,3
    2 46 46 46 46,46,46,77,78,05
    6
6E  370 DATA 79 80,81,80,83,84 8
    5,86 87 88 32,122,123,124
    125 1353
FE  380 DATA 120,32,32,112,113,1
    14 115,116 112,113,32,32,73,
    74,32,54,1280
46  390 DATA 52,32,32 32,32,12,1
    5,7 15,10,32,46,46,46,46,46,
    510
1D  400 DATA 46,46,46 46,32,75,7
    8,75,78 75,78,32,83,84,52,53
    ,1133
7C  410 DATA 54,52,53,53,53 54 5
    2,53,54,52,53,54,75,76,75,78
    ,1415
3A  420 DATA 75,78,32,32,2,1,19,
    45 10,5,12,0,5,6 32,7,376
55  430 DATA 10,1,16,0 5,3,15,32
    ,46,46 48 48,48,46,46,32,460

0F  440 DATA 85 80,87,80 80 32-3
    2 32-100,101,32,32,100 103,3
    2,32 1115
C9  450 DATA 32,154,105,100 32,3
    2 32 107,100 105 110,111,32,
    32 32 32 1116
00  460 DATA 32,80,51,32 32,32,3
    2,13 15,10,5,32,3,8,9,12,400

00  470 DATA 4,5 19 0 32,15,20 2
    1,0,0 32 48,46 46 48,46 486
0C  480 DATA 32,32 32,32,32,20,0
    ,5 15 32,13,5 15 19,1,7,312
77  490 DATA 5,32 3,15,14,20 1,5
    ,14,19,32,49,48,00,00,32,395

53  500 DATA 3,0,1 10 1,3,20,5,1
    8,19,44 2,21,20 32 3,210
73  510 DATA 1,14,32 2 5,32,5,14
    12,1 10 7 5,4,32 20 204

20  520 DATA 15 32 14,15,32 12,5
    ,10 19,32,20,8,1,14 32,54,32
    0
7B  530 DATA 49,52,52,32,3,0,1,1
    0,1,3 20 5 10 19 32 33 346
F3  540 DATA 32,32,32,32,13 3 11
    32,19,21,14,19 32,15 14,32,
    352
E1  550 DATA 73,74,32 54,52,32,1
    ,14,4 32 73 74 32 48 50 56,7
    02
B4  560 DATA 32 48,5,14 32,54,52
    ,32,13,15 4,5,41,32,23,9,407

7C  570 DATA 20,0 32,117,110,32,
    15,10,32,119 120,121,32,40,4
    0,46,922
3A  580 DATA 45,48,48,48,48,48,5
    ,24,10,5 3,20,32,13,15,10,41
    3
00  590 DATA 5 32,10,10,15,7,10,
    1,13 15 32 21,19,5,14,7,248
DA  600 DATA 32 13 3,11,32 5 14
    32,20,0 5 32 14 5,1,10,249
33  610 DATA 32 6 21,20,21,10,5
    32,33,32,32 32,32 32,32 32,4
    12
3E  620 DATA 32,20,0 5 32 13,5,1
    9 19,1,7 5 32,3 15,14,230
66  630 DATA 10 20,10,21 3,20,9,
    15,14,32,11 5 20 32,32,0,275

57  640 DATA 32,49 57,56 58 32,6
    10,1,14 11 32 22,1,14 32 43
    3
C2  650 DATA 20 9 7,7,5 12 5 14
    32,32 32 25 15 21,20 32 252
54  660 DATA 3 15 13 13,15 4 15,
    18 5,39 19 32,19 5 18 9 242
A1  670 DATA 15 21,19 32,21 19,5
    ,10 33 19,32 7 21 9 4,5 206
9F  680 DATA 32,32 32,32,32,32 3
    2 32,32,32 32 32,32,32 32
    ,512
8C  690 DATA 0 0,0,0,0,0,0,0,163
    ,0,133,250 169 145,133,251,1
    254
11  700 DATA 169,0,133,174,133,1
    53 169 9 133 175 133,184,169
    ,0 133,252 2176
08  710 DATA 169 153,133 253 160
    ,0,177,200 145,174,230,250 2
    00,2,230,251,2705
0F  720 DATA 230 174,250,2,230,1
    75,165 250 187,252,200,254,1
    65,251 197,253 3191
56  730 DATA 200,220,160,103,133
    ,107,169 153 133,108,169 4,1
    33,183 169,0 2389
60  740 DATA 133 185 160,0 185,1
    15,153 249,6,32 210,255 200
    250 245 32 2359
3F  750 DATA 207 255 240 251 201
    48,240 4-201,58,48,230,41,1
    5,133,106 2357
C8  760 DATA 76,234,245,147,17,1
    7 73 70 00 00 54 32,00,00 00
    ,73,1464
00  770 DATA 67,69,32 76,85,77,0
    6,69 82 13,17,07,65 03 01 49
    560
EA  780 DATA 32 47,32,60,73,83,7
    5,61,32,50,32,70,82,32,57,50
    ,939
BD  790 DATA 45 32,0,67,00,00,54
    ,0 0,0,0,0 0,0,0,0 400
F3  800 DATA 0,0 0,255,255 255,2
    55,0,0,0,0,0,0,0 0 0 1000
```

Disk Cat

Keep track of all your files with this simple but essential cataloguing system

This double program consists of Disk Cat, which creates a file and File Editor, which allows you to make any changes to it

With Disk Cat you can create a sequential file on disk containing a program title and a disk number, such as PACMAN 27A You will be able to print out an alphabetical double-column master list or small notebook size pages to cut out and keep in a ring file

You will see from your list that PACMAN is on disk number 27, side A. If you do not double your disks then the A or B need not be included in the disk number

Before using Disk Cat, I use a directory logging program to make a printout of all my program titles and allocate a number to each disk.

When running Disk Cat you will be asked to wait while the sort routine is poked into memory When prompted, press the space bar and you will be presented with a menu

If you are creating a new file select option A. The screen will clear and an instruction box will appear at the top of the display This tells you how to close a file – the only program title you can't use is 'END', on disk number '0'

Now enter the title (16 characters maximum), press return and enter the disk number (four characters maximum) When you have entered all the titles and disk numbers, simply type END, press the return key, type 0 for the disk number, press return again and the file will be properly closed

Selecting option B will allow you to extend an existing file The drive will run for a few seconds as it places the read/write head at the end of the file Then proceed as in option A.

Option C will read your file into memory and automatically sort it into alphabetical order and then you will be presented with a new menu

When you choose option D the screen will clear and you'll be asked to enter the date You should have your printer switched on at this stage The date should be entered in the form DD/MM/YY Press return and in a few moments you will have a double column, alphabetical master list of all your program titles and disk numbers Disk Cat has been tested on the MPS 801,803 and MPS 1000.

Option E will display a submenu, select P for printer or S for screen You will then be asked to enter a category If you enter C, every title beginning with C will be listed either to the screen or the printer If you enter two letters, eg CL, then only the titles beginning with CL will be listed

If you originally selected P the output will be sent to the printer and this will give you a 125x90mm page which can be cut out and put into a small ring file

If you make a mistake entering a title or disk number don't panic, you can use File Editor to make any corrections, changes or deletions It will then scratch the old file from disk and replace it with the update File Editor has on-screen instructions and you should find it simple to follow

```
PROGRAM: DISK CAT
3C  10 POKE645,0
DF  20 PRINT"{CLR}" POKE53280,15
      :POKE53281,11 PRINTAB(11)"{
    RVSON}CS CA 5=17,CS}"
0D  30 PRINTTAB(11)"{RVSON,S-,SP
    C17 S-}"
EC  40 PRINTTAB(11)"{RVSON,S-,SP
    C4}DISK CAT{SPC5 S-}"
A1  50 PRINTTAB(11)"{RVSON S-,SP
    C17 S-}"
EE  60 PRINTTAB(11)"{RVSON,S- SP
    C7}BY{SPC8,S-}"
F5  70 PRINTTAB(11)"{RVSON S-,SP
    C17 S-}
0A  80 PRINTTAB(11)"{RVSON,S-,SP
    C3}JOHN  HYDE{SPC4,S-}"
4E  90 PRINTTAB(11)"{RVSON,S-,SP
    C17 S-}"
E0  100 PRINTTAB(11)'{RVSON,C2,S
    =17 CX}"

05  110 RR=13 C=6 GOSUB1000 GOSU
    B650
9S  120 DIMX$(1000):LS="{RVSON,C
    A,5=0 CS}" CS="{RVSON,S8}DIS
    K CAT{S8}" 8S="{RVSON,CZ,S=8
    ,CX}" CR$=CHR$(13)
97  130 SP8="{SPC39}"
FE  140 NAS="TITLE" CNS="DISK#"
    ML8="MASTER LIST" OCS="DISK
    CAT"
46  150 Z=0 FORJ=49152TO49364
41  160 READA POKEJ,A Z=Z+A NEXT
    J
B4  170 IFZ<>29842THENPRINT"ERRO
    R IN DATA!" END
9O  180 POKE45245,9 POKE649,10 R
    R=13 C=8 GOSUB1000 NN=7 GOSU
    B1300
3S  190 RR=16 C=8:GOSUB1000
5O  200 PRINT"{RVSON,CA,9*21,CS}
    "
31  210 PRINTTAB(8)"{RVSON,S-,SP
    C21 S-}"
9O  220 PRINTTAB(8)"{RVSON,S-}KI
    T SPACE TO CONTINUE{S-}"
B5  230 PRINTTAB(8) {RVSON,S- SP
    C21,S-}"
63  240 PRINTTAB(8)"{RVSON CZ,5*
    21,CX}"
F5  250 GETA8 IFA8<>CHR8(32)THEN
    250
21  260 GOTO720
E2  270 OPEN1S,8,15 OPEN2 8 2,"8
    0C TITLE,S,W"
28  280 GOTO310
97  290 OPEN1S,8,15 OPEN2 8,2, "8
    0C TITLE A"
C4  300 RR=11 GOSUB1000
94  310 GOSUB650 PRINTTAB(13)'{R
    VSON CA,5*11,CS}"
6A  320 PRINTTAB(13)"{RVSON,S-}E
    NTER TITLE{S-}"
9O  330 PRINTTAB(13)"{RVSON,CZ,5
    *11,CX,RVSOFF}"SPC(S0) INPUT
    PN$
```

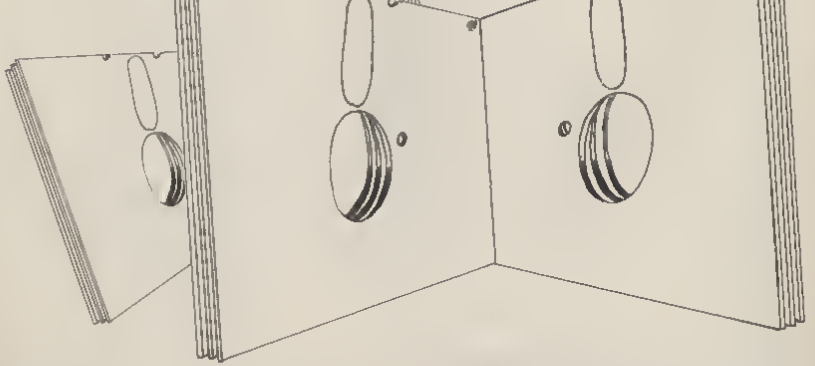
by J R Hyde

```
91  340 G=LEN(PN$):IFS>1GTHENRR=
    11:NN=5:GOSU81500:GOTO1770
C0  350 RR=11:NN=5:GOSU81500
FF  360 PRINTTAB(5)"{RVSON,CA,S*
    17,CS}"
E0  370 PRINTTAB(5)"{RVSON,S-}EN
    TER DISK NUMBER{S-}"
02  380 PRINTTAB(5)"{RVSON,CZ,S*
    17 CX RVSOFF}"SPC(38) INPUTN
    $
32  390 S$=LEN(N$):IFSS>4THENRR=
    11:GOSU81500:GOTO1810
04  400 IFPN$="END"ANDN$="0"THEN
    GOTO440
A0  410 PRINT#2,PN$;CR$;N$
56  420 GOSU8350:NN=13:RR=11:GOS
    U81500
AC  430 IFPN$<>"END"THEN310
33  440 CLOSE2:CLOSE15:GOTO740
7A  450 RR=10:C=0:GOSU81850:PRIN
    TTAB(13)"{RVSON,CA,S=12,CS}"
    SPC(26)
A1  460 PRINT"{RVSON S-}READING
    FILES-1"SPC(28)
D4  470 PRINT"{RVSON CZ,S*12,CX}
    "
DD  480 OPEN15,8,15:OPEN2,8,2,"0
    :DC TITLE S #'
2E  490 GOSU8650:I=0:X=0
C3  500 INPUT#2,PN$,N$
A5  510 NU$="0000"+N$:A=LEN(NU$)
    :IFA>=4THENA=4:N$=RIGHT$(NU$
    ,A)
30  520 RS=ST:GOSU8650
DA  530 IS="{5PC28}"
15  540 D=LEN(PN$):IFQ<-17THEND=
    21-LEN(PN$)
00  550 IS=LEFT$(IS,D)
04  560 I=I+1:X=X+1:X$(I)=PN$+IS
    +N$
35  570 IFRS=0THEN580
6F  580 IFRS<>64THENPRINT"STATUS
    =";RS
0E  590 CLOSE2:CLOSE15
AA  600 RR=10:C=0:GOSU81850:NN=6
    :GOSU81500
8C  610 PRINTTAB(13)"{RVSON,CA,S
    *12,CS}"SPC(26)
EA  620 PRINT"{RVSON S-}SORTING
    FILES-1"SPC(28)
25  630 PRINT"{RVSON,CZ,S*12,CX}
    "
0A  640 GOSU81450:GOSU81500:GOTO
    670
F6  650 INPUT#19,EN,EM$,ET,ES
01  660 IFEN>0THENPRINTEN,EM$,ET
    ,ES:STOP
2E  670 RETURN
FC  680 PRINTCHR$(147)TAB(15)LB:
    PRINTTAB(15)CS:PRINTTAB(19)B
    $
49  690 PRINT"{RVSON,CA S*20,CS}
    "
0A  700 PRINTTAB(6)"{RVSON,S-}IN
    ITIALISING   PLEASE WAIT{S-}
    "
45  710 PRINTTAB(6)"{RVSON CZ,S*
    26,CX}":RETURN
0F  720 PRINTCHR$(147)TAB(99)LS:
    PRINTTAB(15)CS:PRINTTAB(15)B
    :RR=8:C=5:GOSU81850
E5  730 GOTO790
44  740 RR=6:C=0:GOSU81850:NN=10:
    GOSU81510
CE  750 PRINTTAB(8)"{RVSON,CA,S*
    20,CS}"
05  760 PRINTTAB(5)"{RVSON,S-}A)
     CREATE A NEW FILE{S-}"
BD  770 PRINTTAB(5)"{RVSON,S-,SP
    C20,S-}"
A8  780 PRINTTAB(5)"{RVSON,S-}B)
     ADD TO A FILE{SPC4,S-}"
89  790 PRINTTAB(5)"{RVSON,S-,SP
    C20,S-}"
A2  800 PRINTTAB(5)"{RVSON,S-}C)
     READ A FILE{SPC6,S-}"
05  810 PRINTTAB(8)"{RVSON,C2,S*
    20,CX}"
54  820 GETA$:IFA$=""THEN820
28  830 IFA$="A"THENGOSU8350:GOT
    O270
6D  840 IFA$="B"THENNN=10:RR=5:G
    OSU81000:RR=5:GOSU81000:GOSU
    81000:GOTO250
C2  850 IFA$="C"THENRR=9:NN=16:G
    OSU81500:GOTO450
64  860 IFA$<>"A"ORA$<>"B"ORA$<>
    "C"THEN820
E0  870 RR=10:GOSU81850:PRINTTAB
    (6)"{RVSON,CA,S*27,CS}"
98  880 PRINTTAB(5)"{RVSON,S-}01
     PRINT MASTER FILE{SPC7,S-}"
4C  890 PRINTTAB(5)"{RVSON,S-,SP
    C27,S-}"
55  900 PRINTTAB(6)"{RVSON,S-}E)
     PRINT ALAPHABETICAL PAGE{S-
    }"
7B  910 PRINTTAB(6)"{RVSON,S-,SP
    C27,S-}"
30  920 PRINTTAB(6)"{RVSON,S-}Z)
     QUIT PROGRAMME{SPC10,S-}"
54  930 PRINTTAB(6)"{RVSON,CZ,S*
    27,CX}"
18  940 GETA$:IFA$=""THEN940
51  950 IFA$="0"GOTO1060
E1  960 IFA$="E"GOTO1180
C0  970 IFA$="Z"THENCLR:END
12  980 IFA$<>"0"ORA$<>"E"ORA$<>
    "F"THEN940
58  990 PRINTCHR$(147)TAB(55)L$:
    PRINT11A$(15)CS:PRINTTAB(15)0
    $
80  1000 PRINTTAB(5)"{RVSON CA,S
    *27,CS}"
E6  1010 PRINTTAB(6)"{RVSON,S-}I
    D RETURN TO MENU ENTER ENDOS
    -}"
9E  1020 PRINTTAB(6)"{RVSON,S-,S
    PC27,S-}"
C0  1030 PRINTTAB(6)"{RVSON,S-}
    PRESS RETURN  AND ENTER 0 {S
    -}"
F1  1040 PRINTTAB(6)"{RVSON,CZ,S
    *27,CX}"
90  1050 RETURN
5C  1060 CLOSE2:OPEN2,4,LC=55:NN
    =13:RR=10:GOSU81500
12  1070 PRINTTAB(14)"{RVSON CA,
    S*10,CS}"
E4  1080 PRINTTAB(14)"{RVSON,S-}
    ENTER DATE{S-}"
6C  1090 PRINTTAB(14)"{RVSON,CZ,
    S*10,CX}" INPUTDA$
90  1100 GOSU81500:PRINTTAB(13)"
    {RVSON,CA S*13,CS}"
94  1110 PRINTTAB(13)"{RVSON,S-}
    PRINTING FILE{S-}"
10  1120 PRINTTAB(13)"{RVSON,CZ,
    S*13,CX}" GOSU81410
0F  1130 V=INT((X+I)/2)
30  1140 FORJ=1TOV:X=J:X=V+J
C6  1150 PRINT#2,""TAB(7)X$(J) I
    A$(@)X$(X) LC=LC+1
93  1160 IFLC=55THENFORZ=1TO3:PR
    INT#2,CHR$(10) NEXT:GOSUB1410
    @
77  1170 NEXTJ:CLOSE2:GOSUB1500
    :GOTO870
56  1180 NN=8:GOSUB1500:PRINTTAB
    (@)"{RVSON CA S*20 CS}"
5D  1190 PRINTTAB(@)"{RVSON,S-}P
    ) OUTPUT TO PRINTER{S-}"
97  1200 PRINTTAB(@)"{RVSON,S-,S
    PC20,S-}"
40  1210 PRINTTAB(@)"{RVSON,S-}S
    ) OUTPUT TO SCREEN {S-}"
03  1220 PRINTTAB(@)"{RVSON,S-,S
    PC20,S-}"
D4  1230 PRINTTAB(@)"{RVSON,S-}X
    ) RETURN TO MENU{SPC3,S-}"
0F  1240 PRINTTAB(@)"{RVSON,CZ,S
    *20,CX}"
6F  1250 GETA$:IFA$="" THEN1250
CA  1260 IFA$="P"THENGOTO1300
97  1270 IFA$="S"THENGOTO1520
E5  1280 IFA$="X"THENGOTO@@@@
0D  1290 IFA$<>"P"ORA$<>"S"ORA$<
    >"X"THEN1250
C5  1300 RR=10:NN=7:GOSU81500:PR
    INTTAB(6)"{RVSON,CA,S*20,CS}
    "
70  1310 PRINTTAB(6)"{RVSON,S-}E
    NTER CATAGORY TO PRINT  A-Z{
    S-}"
F6  1320 PRINTTAB(5)"{RVSON,CZ,S
    *20,CX}"
0B  1330 INPUTCA$
28  1340 CLOSE3:OPEN3,4 LC=@:GOS
    UB1470
8C  1350 FORJ=1TOI
F4  1360 IFCA$=LEFT$(X$(J),1)THE
    NPRINT#3,""TAB(3)X$(J),TAB(4
    )CHR$(I@4) LC=LC+1
@C  1370 IFLC=22IXENGOSUB1420
35  1380 NEXTJ
AA  1390 FORZ=LC1026:PRINT#3,""T
    AB(32)CHR$(I@4):NEXTZ
CE  1400 FOR1=1TO32:PRINT#3,CH@@
    (196)  NEXTI
5C  1410 PRINT#3:CLOSE3:GOSUB150
    @:GOTO870
8E  1420 FORZ=LCTO26:PRINT#3,""T
    AB(32)CHR$(I@4):NEXT
2D  1430 FORI=1TO32:PRINT#3,CHR@
    (155):NEXT:PRINT#3,CHR$(I@)
    :GOSUB1470:RETURN
FD  1440 PRINT#2,""TAB(6)CHR@(14)DC@,TAB(3)ML@,TAB(3)DA$:PRINT#2
G1  1450 PRINT#2,""TAB(4)NA@,TAB(4)DN$:TAB(3)NA$,TAB(4)DN$:CHR$(15):PRINT#2:LC=@
37  1460 RETURN
4C  1470 PRINT#3,CH@@(14)TAB(2)NA$,TAB(3)DN@,CHR$(15)TAB(2);CHR$(14);:PRINT#3
2A  1485 LC=@:RETURN
@@  1490 N2=INT(I/256):N1=I-256*N2
21  1500 POKE49350,N1:POKE49367,N2
AA  1510 SYS49152:RETURN
7E  1520 RR=5:NN=19:GOSUB1500:P@
```

```
    1NITIA0(1991"CRVSON,CA,5*28 C
    SJ"
43  1930 PRINTTA0(51"CRVSON,S-JE
    NTER CATASORY TO PRINT: A-Z(
    S-J"
7D  1940 PRINTTA0(53'CRVSON,C2,R
    *28,CX,DOWN)"
05  1950 1NPUTCA$
1A  1960 RR-S OOSU01000 LC-0 L-L
    EN(CA$1 Z-ASC(LEFT0(CA$,1))
1F  1970 FORJ-1TO1
07  1980 1FCA0-LEFT0(X$(J1 L)THE
    NPRINTTA0(0)X$(J):LC-LC+1
E1  1990 1FA$C(LEFT0(X0(J1,111>Z
    THENJ-1
E0  1000 IFLC-14THENGOSU01710
1C  1010 NEXTJ
50  1020 PRINTTA0(47)"CRVSON,CA,
    S*24 CSJ'
75  1030 PRINTTA0(71"CRVSON,S-JP
    RESS RETURN TO CONTINUECS-J"
6A  1040 PRINTTA0(7)"CRVSON,S-,S
    PCSJOR X FOR MENUCSPC0 S-J"
    POKE150,0
98  1050 PRINTTA0(71"CRVSON,CZ,S
    *24,CX)" POKE198,0
10  1060 GETA$ IFA$-""THEN1060
E5  1070 IFA$- X"THENRR-5 GOSU01
    900 GOTO870
A5  1080 1FA$<>CHR$(131THEN1020
29  1090 1FA$<>CHR$(13)ORA$<>CHR
    $(85)THEN1060
A3  1700 GOTO1020
7A  1710 PRINTTA0(47)"CRVSON,CA,
    S*24 CSJ"
2F  1720 PRINTTA0(7)"CRVSON,S-JP
    RE55 RETURN TO CONTINUECS-1"
20  1730 PRINTTA0(7)"CRVSON CZ,S
    *24 CXJ"
0F  1740 GETA$ 1FA0-""THEN1740
99  1750 1FA$<>CHR$(131THEN1740
FC  1760 GOSU01500 LC-0 RETURN
F4  1770 PN$-"" PRINTTA0(1("CRVS
    ON CA,S*35 CSJ"
45  1780 PRINTTA0(11"CRVSON,S-JT
    ITLE MUST 0E 10 CHARACTERS O
    R LESSCS-J"
0F  1790 PRINTTA0(1)"CRVSON,C2 S
    *35,CXJ" OOSU01050 NN-11
E1  1800 GOSU01500 GOTO310
69  1810 NS-"" PRINTTA0(1)"CRVSO
    N,CA,S*36 CSJ"
70  1820 PRINTTA0(1)"CRVSON,S-30
    1SK N0 MUST 0E 4 CHARACTERS
    OR LESSCS-J"
C0  1830 PRINTTA0(1)"CRVSON,C2,S
    *36 CXJ" GOSU01050 NN-11
5E  1840 GOSU01500 GOTO300
1C  1850 FORN-5427201054286 POKEA
    ,0 NEXTA
51  1060 POKE54278,4 POKE54273,4
    0 POKE54277 0 POKE54270,240
    POKE54296,15
F3  1870 FORZ-1TO10 POKE5427G,17
    POKE54276,16 FORU-1TO200 NE
    XTU NEXTZ
CC  1000 POKE54296,0 RETURN
DA  1050 R-RR FOKE7M1 R POKE7M2,
    C POKE703,0 SYS65520 RETURN
F3  1500 C-0 GOSU01890
57  1910 FORP-1TONN PRINTSP$ NEX
    T
C7  1920 C-0:GOSU01890:RETURN
F7  1930 DATA173 214 152,200,6,1
    73 215 152,200 1,36,155,47,2
    4,105 10 133 34,155
F0  1940 DATA48 105 0,133,35,76,
    153 192,160,0,177,34,72 177,
    36,145 34 104 145 36
99  1950 DATA200 192 3,200 241,7
    6 116 152,169,0,141,218,192
    189,177,34 240 59
30  1960 DATA141,213 192 177 39
    240 219 205 213 192 179,0,14
    1,213,192 169 1,141
30  1970 DATA218 192,200 177,34,
    133,251,177,36 133 253 000,1
    77,34 133 252 177 36
05  1980 DATA133,254 160 0,177,2
    53 205 251 144 100 200 11 20
    0 206 213 192 200
A0  1990 DATA242 173 218 192 200
    ,107 165 38 24 105,3,133,36
    165 37 105,0 133 37
21  2000 DATA173 216,192,200,3,2
    06 217 192 206 219,192,173,2
    16 192 200 152 173
2C  2010 DATA217 192 200,153,185
    34 24 105 3 133 34,165 35,1
    05 0,133 35 173,214
69  2020 DATA152 200 3,206 215,1
    92,206 214 152 173,214,192,2
    00,6,173 215 192 200
10  2030 DATA1,96 173,214,192 14
    1,216 192,173,215 192 141,21
    7 152,165 34 24 105
02  2040 DATA3,133,36 165,35,105
    ,0,133,37,76 47 192
```

```
PROGRAM: DISK CAT 2011GR
C9  10 PRINT"[CLR]" POKE53280,15
    POKE53281,11
3E  15 RR=2 C=13 GOSU07000 PRINT
    "[C@ RVSON CA,S*11 CS]"
BC  20 PRINTTAB(13)"CRVSON S-JFI
    LE EDITORCS-J"
14  25 PRINTTAB(13)"CRVSON,S-,SP
    C11 S-J"
32  30 PRINTTAB(13)"CRVSON,S- SP
    C4)0YCSPCR,S-J"
49  35 PRINTTAB(13)"CRVSON,S-,SP
    C11 S-J'
AA  40 PRINTTAB(13)"CRVSON S-J J
    OHN HYDE (S-J"
48  45 PRINTTAB(13)"CRVSON,CZ,S*
    11,CX)"
0D  50 REM: [illegible]
    )
42  100 CR$=CHR$(131:SP$="[SPC39
    J"
7D  110 LR="[CRVSON,CA S*11,CS]"
    TS-"CRVSON,S-JFILE EDITORCS-
    J" 0R-"CRVSON,CZ S*11,CX)"
02  120 DIMPR$(10001 DNR(10001
10  150 GOSU04000 OOSU0300 RR-10
    :C-13:GOSU07000 PRINT"CRVSON
    ,CA,S*12,CS)"
9A  101 PRINTTA0(13)"CRVSON,S-JR
    EADIND FILECS-J"
73  102 PRINTTA0(131"CRVSON CZ S
    *12 CX)"
5E  200 CLOSE15 OPEN15,0,15 OPEN
    2,0 2,"0-DC TITLE,S,R"
35  210 GOSUB1000 I-0
1A  220 INPUT#2 PN$,N$
9C  230 R9-ST GOSUB1000
7F  240 I-I+1 PR$(I)-PN$ DN$(I)-
    N$
6D  250 IFR9-0THEN220
3A  260 IFR$<>"THENPRINT"STATUS
    -",R$
E4  270 CLOSE2 CLOSE15 RR-9 NN-5
    GOSUB5000 GOTO20000
71  300 PRINTCHR$(147) RR-1 C-13
    GOSU07000 PRINTLR
2A  310 PRINTTAB(13)TR
02  320 PRINTTAB(13)0$ RETURN
01  000 RR-0 C-4 GOSU07000 PRINT
    "CRVSON CA,S*30,CS)"
0A  501 PRINTTAB(4)"CRVSON,S-)EN
    TER TITLE YOU WISH TO CHANGE
    CS-1"
46  502 PRINTTAB(41"CRVSON,CZ,S*
    30 CX)" CAR-""
0A  510 INPUT"(DOWN)",CA$
A8  RIR IFCAR-""THENRR-7 GOSU850
    00 OOTO2000
03  530 FORJ-1TOI
C0  540 IFPR(J1-CA$THENRR-0 NN-
    5 GOSU05000 RR-R OOSU07000 O
    OTO0000
F4  550 NEXTJ
5C  560 1FCA$<>PR$(J)TKENRR-12:C
    -11 GOSU07000 PRINT"CRVSON,C
    A,S*15,CS)"
74  562 PRINTTAB(11)"CRVSON,S-JT
    ITLE NOT FOUNDCS-J"
50  565 PRINTTAB(11)"CRVSON,CZ,S
    *15 CX)"
E7  570 FORCD-1TO1200 NEXTCD RR-
    5 NN-15 GOSU59000 GOTO2000
60  580 FORR-1TO3W PRINTTAR(1)CH
    R$(1521, NEXTA
62  600 RR-0 C-2 GOSU07000 PRINT
    "CRVSON,CA,S*35 CS)"
0E  001 PRINTTAB(21"CRVSON,S-)UP
    ARROW RETAINS TITLE OR D15K
    N0 CS-J'
35  602 PRINTTAB(2) CRVSON,S-,SP
    C35,S-J"
0E  010 PRINTTA0(2)"CRVSON,S-)0A
    CK ARROW DELETES TITLE 0 D1S
    K N0 CS-J"
4B  011 PRINTTAB(2)'CRVSON,S-,SP
    C3R,S-J"
00  012 PRINTTA0(2)"CRVSON S-,SP
    C10)RETURN TO A0DRTCSPC10 S-
    J"
02  613 PRINTTA0(2)"CRVSON,CZ,S*
    35,CX)"
44  015 REMPN$-"":NR-""
0F  020 PN$- INPUT"(DOWN) PN$
    -S-LEN(PN$) IFR>10THENSOTO59
    00
FD  625 1FPN$-CHR$(34)THENPN$-PR
    R(J):PRR(J1-PR$(J)
D9  626 1FPN0-""THENPRR(J1-PR$(J
    1 DNR(J)-DN$(J) J-J+1 RR-5 N
    N-10 GOSU05000 OOTO2000
3F  630 1FPN$-"_"THENPRR(J)- ' O
    NR(J)-"":NN-10:GOSU05000 OOT
    O2000
E0  635 RR-10 NN-9 GOSU59000
```

```
07  640 RR=10 PRINTTAB(5)"[RVSON
    ,CA,S*17,CS]"
3D  641 PRINTTAB(5)"[RVSON,S-]EN
    TER DISK NUMBER[S-]"
E4  642 PRINTTAB(5)"[RVSON,CZ,S*
    17,CX]'
05  650 N$="";INPUTN$:SS=LEN(N$)
    :IFSS>4THENGOTO5010
33  651 IFN$=CHR$(54)THENN$=DN$(
    J):DN$(J)=DN$(J)
53  656 IFN$="" THENPR$(J)=PR$(J)
    :DN$(J)=DN$(J):J=I+1:RR=5 NN
    =16:GOSUB5000:GOTO2000
43  680 PR$(J)=PN$:DN$(J)=N$:J=I
    +1:RR=5:NN=10:GOSUB5050:GOTO
    2000
0F  1000 REM:      CHECK ERROR CH
    ANNEL,,,,
E3  1010 INPUT#15,EN,EM$,ET,ES
E4  1020 IFEN>0THENPRINTEN,EM$,E
    T,ES:STOP
04  1030 RETURN
05  2000 RR=9:C=6:GOSUB7000:PRIN
    T"[RVSON,CA,S*25,CS]"
04  2010 PRINTTAB(6)"[RVSON,S-]A
    ) CHANGE OR DELETE TITLE[S-]
    "
C7  2011 PRINTTAB(6)"[RVSON,S-,S
    PC25,S-]"
6D  2020 PRINTTAB(6)"[RVSON,S-]B
    ) SAVE CHANGES[SPC]0,S-]"
55  2025 PRINTTAB(6)"[RVSON,S-,S
    PC25,S-]"
0)  2030 PRINTTAB(6)"[RVSON,S-]C
    ) QUIT PROGRAMME[SPC]0 S-]"
73  2031 PRINTTAB(6)"[RVSON,CZ,S
    *25,CX]"
0C  2040 GETA$:IFA$=""THEN2040
7C  2050 IFA$="A"THENRR=7:NN=10
    :GOSUB5000:GOTO500
0F  2060 IFA$="B"THENRR=7:NN=10
    :GOSUB5000:GOTO3000
DF  2070 IFA$="C"THENPRINTCHR$(1
    47):CLR:END
57  2080 IFA$<>"A"ORA$<>"B"ORA$<
    >"C"THEN2040
42  3000 REM | | SAVE UPDATED F
    ILE
A0  3005 RR=11:C=5:GOSUB7000:PRI
    NT"[RVSON,CA,S*15,CS]"
12  3010 PRINTTAB(5)"[RVSON,S-]S
    AVING UPDATED FILE[S-]"
50  3013 PRINTTAB(5)"[RVSON CZ,S
    *15,CX]"
77  3015 OPEN15,8,15,"S0:DC TITL
    E"
E7  3020 OPEN2,8,2,"0 DC TITLE,S
    ,W"
51  3030 FORJ=1TO1:PRINT#2,PR$(J
    ):CR$;DN$(J):NEXTJ
C2  3040 GOSUB1000:CLOSE2:CLOSE1
    5:GOSUB5000:GOTO2000
41  4000 RR=12:C=7:GOSUB7000:PRI
    NT"[RVSON,CA,S*24,CS]"
C0  4001 PRINTTAB(7)"[RVSON,S-]P
    UT FILE DISK INTO DRIVE[S-]"
54  4005 PRINTTAB(7)"[RVSON,S-,S
    PC24,S-]"
75  4010 PRINTTAB(7)"[RVSON,S-]H
    IT ANY KEY TO CONTINUE [S-]"
06  4011 PRINTTAB(7)"[RVSON,CZ,S
    *24,CX]"
C9  4020 GETA$:IFA$=""THEN4020
43  4030 RETURN
04  5000 RR=6:NN=5:GOSUB3000:PR
    INTTAB(2)"[RVSON CA S*35 CS]
    "
4H  5001 PRINTTAB(2)"[RVSON,S-]T
    ITLE MUST BE 16 CHARACTERS O
    R LESS[S-]"
FC  5002 PRINTTAB(2)"[RVSON,CZ,S
    *35 CX]" PN$="'" :GOSUB5020
CC  5003 GOSUB3000:GOTO620
FD  5010 GOSUB5000:PRINTTAB(I)"[
    RVSON,CA,S*35,CS]"
34  5011 PRINTTAB(1)'[RVSON,S-]D
    ISK NO MUST BE 4 CHARACTERS
    OR LESS[S-]"
1S  5012 PRINTTAB(1)"[RVSON,CZ,S
    *30 CX]" N$="" :GOSUB5020
00  5015 GOSUB3000:GOTO640
61  5020 FORA=54272TO54296:POKEA
    ,0:NEXT
13  5030 POKE54272,4:POKE54273,4
    0:POKE54277,0:POKE54278,245
    :POKE54296,15
2S  5040 FORZ=1TO10:POKE54276,17
    :POKE54276,16:FORV=1TO150:NE
    XTV:NEXTZ
FA  5050 POKE54296,0:RETURN
0G  6000 RR=17:C=30:GOSUB7000:FO
    RZ=1TO117:PRINTCHR$(20),:NEX
    T
20  6010 RR=15:C=0:GOSUB7000:RET
    URN
SE  6100 RR=22:C=0:GOSUB7000:FOR
    Z=1TO10:PRINTCHR$(20),:NEXT
0E  6110 RR=15:C=0:GOSUB7000:RET
    URN
G4  7000 R=RR:POKE781,R:POKE782,
    C:POKE783,0:SYS65520:RETURN
CE  8000 DEF FNC(X)=20-(LEN(TE$)
    /2)
BC  8010 TE$=PR$(J)+"  "+DN$(J):
    PRINTTAB(FNC(X)),TE$:GOTO580
0B  9000 C=0:GOSUB7000
7F  9010 FORP=1TONN:PRINTSPS:NEX
    T
F0  9020 C=0:GOSUB7000:RETURN
```



Program Compactor

Reduce long files and link up your programs to save on disk space

by Richard Bright

After a game has been written there are normally several chunks of code which are spread about in the computer's memory. The Program Compactor will help to tie these routines together and then crunch them down into a neat little file.

The program will not work on Basic programs but it will happily attack anything lying from $0300 to $FFFF. Any number of files can be linked as long as the total area covered does not exceed 234 disk blocks.

When the program runs, it fills up the program memory with zero bytes to help the cruncher to do its work. When finished a prompt appears asking for LO-MEM to be input. The lowest memory location occupied by any part of the program code is entered in hex, as are all inputs in the program.

A SKIP value is the next request which means that the cruncher will ignore the 768 bytes following the input value. This is useful in skirting around screen areas. For example, if code starts at $0300 and another piece of code starts at $0800, the screen area can be omitted by entering $0400 at the SKIP prompt.

The next job is to load the first file into the computer. At the FILE-NAME prompt, entering '$' will display the disk directory. When the program name is known, it can then be entered (without quotes) at the name prompt.

When the program has loaded, the rest of the files can be loaded until everything is in memory. Now

it's crunch time and entering "" instead of a filename will set the process into motion.

A screen printout of memory space used and the current disk block rating is given while the border flashes to indicate that the crunch is proceeding. After the cruncher comes the optimiser which further compresses the data.

On completion, the program prints the new memory usage and block count details before asking for the hex boot location for the coded program. Next a filename for the saved program is needed and this is the last point at which disks can be changed. Enter the new filename, press return and the job's complete.

Another copy can now be saved with a new filename, if necessary, or pressing the return key without entering anything will reset the program ready for a new compacting session.

If anything goes wrong at any point the program can be restarted by pressing RESTORE. This will reset all the program parameters and take the program back to the opening screen.

In tests a 232 block program was reduced to 131 blocks – a saving of 101 blocks which corresponds to 25687 bytes! As you can see the compactor really does save on disk space.

```
PROGRAM  COMPACTOR.BAS

C1   10 REM TYPE IN THE PROGRAM A
     ND SAVE IT ON A SPARE CASSET
     TE OR DISK
30   20 REM PREPARE A DIFFERENT D
     ISK OR CASSETTE FOR SAVING T
     HE PROGRAM
9D   30 REM THE PROGRAM SAVES AFT
     ER ASKING FOR A DEVICE NUMBE
     R.  SO BEWARE!!!
4E   40 BL=200   LN=50   SA=3584
     4
3E   50 FOR L=0 TO BL CX=0 FOR D=
     0 TO 15 READ A CX=CX+A
31   60 POKES3280,A POKE SA+L*16+
     D,A NEXT D
80   70 READ A IF A><CX THENPRINT
     "ERROR IN LINE",LN+(L*10) ST
     OP
1C   80 NEXT L SY540024
A6   90 DATA 20,8,185,7,150,50,48
     ,55,52,32,84,86,82,66,75,45,
     1066
0C   100 DATA 58,55,0,0,0,170,170
     ,170,170,32,88,229,189,14,14
     1,32,1475
EF   110 DATA 208,141,33,208,120,
     185,52,133,1,162,5,189,84,8,
     157,45,1725
35   120 DATA 0,202,16,247,154,18
     0,0,169,50,208,65,0,177,45,1
     53,0,1695
28   130 DATA 0,230,1,141,24,212,
```

```
    150,1,200,206,241,165,50,201
,3,208,2088
A0  140 DATA 230,105,100,8,153,0
,1,200,208 247,76,0,1,0,3,5,
8E  150 DATA 245,84,20,177,47,42
,42,42,42,41,7,170,196 26,1,
141,1316
96  160 DATA 24,1,177,47,41,31,1
70,32,34 1 76,255 1,168,74,1
75,1307
A1  170 DATA 125,92,96,70,48,230
,1,238,32 200,198,1,230,47,2
06,2,1796
54  180 DATA 230,48,96,177,47,32
,34,1,145,45,230,45,208,2,23
0,46,1616
5A  190 DATA 202,208,245,240,130
,169,0,240,239 169 255,208,2
35,177,47,145,2869
88  200 DATA 45,32,34,1,230,45,2
06,2,230,46,202,208,240,240,
164,32,1959
F6  210 DATA 117,1,177,47,32,34
1,145,45,230,45 208,2,230,46
,202,1962
77  220 DATA 208,245,198,57,16,2
41,48,139,134,57,177,47,170,
76,34,1,1848
3E  230 DATA 32,117,1,177,47,145
,45,32,34,1,230,45,208,2,230
,46,1382
58  240 DATA 202,208,240,198,57,
16,236,76,0,1,32,222,1,169,5
5,133,1845
4E  250 DATA 1 88 234,234,234,32
,85,166 76,48,3,224,0,240 23
5,169,2073
FC  260 DATA 3 44,169,8 133,255,
177,47,145,45,200,196,255,20
8,247,24,2156
E8  270 DATA 185,45,101,255,133,
45,165,46,105,0,133,46,160,0
,202,208,1809
8D  280 DATA 225,24,165,47,101,2
55 133,47 165,48,105,0,133,4
8 76,0,1576
2F  290 DATA 1,185,0,239,153,0,2
55,200,208,247,206,224,1,206
,227 1,2353
94  300 DATA 173,227,1,201,47,20
6,234,96,45,138 227,141,223,
124,165,26,2277
88  310 DATA 167,228,167,134,174
,55,162,88,134,49,76,72,170,
0 45,234,1875
76  320 DATA 159,223 71,254,74,2
43,145,242,14 242,60 242,51,
243,87,241,2611
92  330 DATA 202,241,237,246,62,
241,47,243,102 254,165,244,2
37,245,120,162,3046
54  340 DATA 255,154,169,54 133,
1,169,52,141,24,3,169,3,141,
25,3,1496
43  350 DATA 169,0 141,32,208,14
1,33 208,160,0,185,98,3,240,
6,32 1653
66  360 DATA 210,255,200,208,245
,88 78,0,9,152 147,6,14,148,
155,82 2035
3A  370 DATA 95,78,57,72,47,204,
73,78 75,69,82,32,214,49,46,
50,1321
3D  380 DATA 13,13,87,82,73,84,8
4,65,76,32,66 89,32,212,85,6
2,1181
37  390 DATA 68,79,32,105 82 65
67,75,69,82,45,56,55,13,13 1
69,1157
2A  400 DATA 35,133,1,96 227,234
,96,83,32 69,166,76,174,167,
224 0,1825
83  410 DATA 240,235,169,3,44,16
9,8,133,255,177,47,145,45,20
0,196,255,2321
7A  420 DATA 208,247,24,165,45,1
01 255,133 45,165,46,105,0,1
33,46 130,1876
14  430 DATA 0,202,208,229,24,16
5,47,101,255,133,47,165,48,1
05,0,133,1662
DC  440 DATA 48 76 0 1,165 0,239
,153,0,255,200,200 247 206 2
24,1,2043
E3  450 DATA 206,227,1,173,227,1
,201,223,208 234,96,172,35,7
7,79,78,2230
EB  460 DATA 132,1,0,50 20 8,195
,7,168,50,48,55,52,32,64,85,
577
F6  470 DATA 82,66,79,45,56,55 1
63,228,170,98,36,32,60,229,1
65,14,1550
2A  480 DATA 141,32,208 141,33,2
00,120 169,52,133,1,102,5,18
3,54,0,1656
CE  490 DATA 157,45 0 202,16,247
,154,160,0,196,50,206,65 8,1
77,49,1734
72  500 DATA 153,162,60,230,1,14
1,24,212 198,1,200,208,241,1
65,50,201,2247
18  510 DATA 8 200,230,185,100,8
,153,0,1,200,208,247,76,0,1,
199,1823
9C  520 DATA 34,177,47,228,42,56
,153,41,7,170,189,26,1,141,2
4 1,1377
82  530 DATA 177,47,41,31,170,32
,34,1,76,255,1,160,74,175,12
5,52,1459
C3  540 DATA 66,70 48,230,1,238,
32,208 196 1,230,47,208 2,23
0,48,1057
4D  550 DATA 96,177,47,38,34,1,1
45,45,230,45,208 2,230,46,20
2,208,1746
E1  560 DATA 245,240,130,169,0,2
40,238 169 255,208,235,177 4
7,145,45,32,2938
64  570 DATA 34,1,230,45,208,2,2
30,46 202,208,240,240,164,32
,117 1,2000
C2  580 DATA 177,47,32,34,1,145,
45 230,45,208,2,230,46 202,2
06,245,1887
71  590 DATA 198,57,16,241,48,13
9,134,57,177,47,170 76,34,1,
32,117,1544
1A  600 DATA 1,177,47 145,45,32,
34,1,230,45,208,2,230,46,202
,208,1653
41  610 DATA 240,196,57,16,236,7
6,0,1,44,222,1,169 55,76,10,
9,1410
31  620 DATA 167,96,84,120,216,1
69,52,133,1,130,0,132,96,169
,10,133,1735
25  630 DATA 89 152,141,5,3,141,
6,3,141,3,3,145,88 200,208,2
51,1603
33  640 DATA 230,85,208,247,169
55 133,1,88,162,152,130,15,3
2,72,15,1628
30  650 DATA 178,247,142,3,3,140
,4,3,142,54 6,140,65,6,182,2
27 1806
4A  660 DATA 180,15,32,72,15,176
,19,140,6,3,185,255,141,7,3,
152,1379
D2  670 DATA 24,105 15,141,237,3
,169 32,206,7,169,0,141,7,3,
169,1436
16  680 DATA 44,141,251,6,32,83,
10 32,2,14,162,203 160,15 32
,55 1244
4A  690 DATA 15 169,212,32,210,2
55,120,169,52,133 1,173,5,3,
133 50 1778
9B  700 DATA 173 6,3,133 51 169
255,133,88 133,60,160,0,177,
50 145 1851
4C  710 DATA 88 165 88 208,2 198
,88 198 88 165,50,208 2 198
51,158,2076
C4  720 DATA 50 165,51,201 15 20
8 230,230 88 208,2,230 89,15
3,55 133,2204
78  730 DATA 1 88 169 155 32 210
,255,32 147 10,169,13 32 210
,255,165 1583
67  740 DATA 50 141,5,3 141,86 3
,56,233,255,141,58,3,165,51
141,1678
98  750 DATA 6,3,141 87,8 233,7
141,88,8,56 173,86,8 233 88
1403
C3  760 DATA 141 86,8,173,87,8,2
33 17,141,87,8,56 169,0 237,
86,1577
33  770 DATA 8,141,86,8,169,0 23
7,87,8,141 87 8,32 2,14 162,
1220
9F  780 DATA 135,160,15,32 72 15
,142 162,3 140,169,3 160 0 1
85,0,1367
6E  790 DATA 8,153,0 16 200 208,
247,160 88,185,152,3,153,0 1
7,138,1726
EB  800 DATA 16,247 162 216,160
15,32,55,15,32,107,14 173,14
3,14,240,1647
2F  810 DATA 24,165,8,170 160 1
32,186,255,173,145 14,162,15
0,160,14,1667
C4  820 DATA 32 189,255,32,154,1
3,76 41,10,76,234 15,162,169
,160,15,1633
C9  830 DATA 32,55 15 162,216,16
0,15,32 55 15,32,107,14,173
150,14,1247
80  840 DATA 201,36,208,6,32,166
,14,76 90 10 201,33,206 6,17
3,9,1469
A5  850 DATA 3,240,224,96 169,6,
170,160,0,32,186,255,173,145
,14,162,2047
A9  860 DATA 150,160,14,32 189,2
55,32,208,12,76,90,10,120,18
9,52 133,1703
26  870 DATA 1,169,69,133,90,160
```

```
    ,0,169,17,133,91,132,112,230
    ,1,238,1765
25  000 DATA 32,208 158 1,165,11
    2 240_3 76 100,12,177,00,200
    ,205 00 1917
7F  030 DATA 240,34,162,1,165,00
    ,72 165 05,72,160,0,132,113,
    32 161,1746
13  500 DATA 11,240,31,177 00,20
    0,209 00,208,13,201,0,240,20
    ,201,255,2162
90  510 DATA 240,15,200 209 00 2
    40,11,32,40,12,232,224,31,20
    8,223,76,2082
4F  520 DATA 170,11,104,133,00,1
    04,133,00 160,0,224,1,208,10
    ,177,00,1700
41  530 DATA 201,0 240,28,201,25
    5,240_24,138,9,32 32,13,12,1
    77 00,1550
C0  540 DATA 32,13,12,32,161,11,
    202,208,245,185,00,208,144,7
    6 24 12 1634
5A  550 DATA 201,0,240 9,201,255
    ,240,5,200,208,00 206,145,16
    2,1,32,2200
9A  560 DATA 161,11,240 11 209,0
    8,208,7,232,224,31,208,242,2
    40,44,134,2250
E0  570 DATA 32 201,0,208,6,169,
    150 5 92 208,19,201,255,208,
    6,169,1555
FE  500 DATA 192 5,92,208,8,72,1
    65,224,5,92 32,13,12,104,32,
    13,1274
24  590 DATA 12,165,00,240,3 76
    164 10 76,24,12,134,32,162,0
    ,134,1393
E0  1000 DATA 33,72,32,161,11,24
    0 28,104,208,00,72 208_22,16
    5,92,208,1805
7A  1010 DATA 2,230,53,230,52,16
    5 53,201,31,208,231,165,52,2
    01,255,208 2437
47  1020 DATA 225,198,52,165,53,
    9,120,32 13,12,165 52,32,13,
    12,104,1305
AF  1030 DATA 32,13,12,165,00,24
    0,122 76,164,10,160,0,230,00
    ,208,2 1611
4F  1040 DATA 230,00 56 134 52 1
    34,113,162 0,134,53,32,161 1
    1,240,45 1765
72  1050 DATA 177 00 200 209,00
    208,13,201,0,240 34,201,255,
    240 30,200,2384
02  1060 DATA 209,00,240,25 32,4
    0,12 165,92,200,2,230,93 230
    ,92,165 1523
0D  1070 DATA 93 201,31,208,214,
    185,52,201,255,209,200,160,9
    2,165 53,9,2433
1C  1080 DATA 96 32,13,12,165,92
    ,32,13 12 104 133,00,104,133
    ,00 177 1295
1B  1090 DATA 00 32,13,12,32,161
    11 100 52 200,244,100,93 16
    ,240 165 1803
4A  1100 DATA 00,240,14,76,164,1
    0,160,0,145,50,230,50,200,2,
    230,51 1039
FC  1110 DATA 96,169,0,32,13,12,
    165,55,133,1,00,105 11,141,3
    2,208,1325
02  1120 DATA 96,165,00,201,255,
    208 1,96 169 0 32,58,12,169,
    3 32 1554
6E  1130 DATA 58,12,96,133,115 1
    33,115 10,133,117,160,0,132,
    114,164,114,1607
0F  1140 DATA 177,00,200,132,114
    ,164,115 209 00,240,1,96 200
    ,132,115 196,2267
BA  1150 DATA 117,200,235,169,25
    5,133 112,228,104,96 54,165,
    113,240,3,76,2348
09  1160 DATA 220,11,76,233,10,1
    62,1,160,0 138,114,132,112,1
    64,115,132 1703
2F  1170 DATA 115,165,116,133,11
    7,164,114,177,00,200 132,114
    ,164,115,209 00 2211
FD  1180 DATA 208,12,200,132,115
    ,00,117,208,238,232,224,31,
    208,227,138 164,2650
3A  1190 DATA 116,192,0,200,2,9,
    64,32 13,12,177,00,132,115,3
    2,13,1213
1E  1200 DATA 12,164,115,200,196
    ,116,200,242,160,0,138,169 1
    16,224,9,240,2305
F8  1210 DATA 0 133,255,10,24,10
    1 255,208,3,227,10,96,210,24
    ,101,00,1753
6F  1220 DATA 133 00 165,00,105,
    0,133,00,76,164,10,162,142,1
    60,13,32,1561
70  1230 DATA 55,15,32,144,255,1
    69,0,32,152,255,162,8,32,188
    255,32,1845
31  1240 DATA 90,13,133,00,32,90
    ,13,133,00,173,7,9,240 11 16
    5,00,1375
4G  1250 DATA 205,0,5,144,4,233,
    16 133 00,24,165,00,105,16,1
    33,85,1462
8E  1260 DATA 56,165,00,237,3,9,
    133,00,165,00,237,4,9,133,00
    ,32,1537
B5  1270 DATA 90 13,120,160,52,1
    32,1,160,0,145,00,160,55 132
    ,1,230,1535
4A  1280 DATA 00 208,2,230 00,36
    ,144,00,230,32,115,13,165,00
    ,208 2 1730
F0  1290 DATA 150,00,100,00 173,
    5,9,197,00,176 19 208 7 173,
    5 9 1644
6A  1300 DATA 197,00,176,10,165,
    00 141 5,9,165,00,141,5,9,16
    5,255,1713
A2  1310 DATA 141,5,9,96,32,19,2
    30 170 165,144,74 74,176,2,1
    32,96 1583
94  1320 DATA 104,104,00,32,115,
    13 162 123 160,13,76,55,15,1
    00,0 32 1265
01  1330 DATA 195,255,76,231,255
    ,35 35 32,196 73,83,75,32 19
    7 82 82 1534
9A  1340 DATA 79,82,83,32,35,35,
    13,0,210,69,65,60 73 70 71,2
    27,1220
9C  1350 DATA 46 97,60,13,0,162,
    247,160,13,32,55,15,32,143,2
    46,120,1447
00  1360 DATA 169 57,133,165,32,
    213,243,165,106,32,12,237,16
    5,100 32 185 2271
F4  1370 DATA 237,169,1,133,00 1
    69 15 133,05,169 1 32,221,23
    7,165,0,1872
94  1380 DATA 32,221,237,120,160
    ,52,132,1,160,0,177,60,160,5
    5 132,1,1720
B4  1390 DATA 32 221,237,165,00,
    205,5 9,208,7 165 00,205 6,9
    ,240,1891
CE  1400 DATA 0 230 00 208,222 2
    30 05 208,210,60 32,93,245 1
    60 13,76 2188
5F  1410 DATA 210,255,215,82,73,
    84,73,70,71,46 46,13 0 162,7
    0,160,1630
D2  1420 DATA 14,32,00,15,173,9,
    9,133,136,56,173,6,9,233,16,
    133,1196
44  1430 DATA 137,100,136,32,205
    ,100 102,00,160,14,32,55,15,
    160,1,165,1717
08  1440 DATA 137,240,16 200 24,
    165,136,233,253,133,136,165,
    137 233,0,133,2341
F2  1450 DATA 137 208,236,152,17
    0,169 0,32,205,100 162 96,16
    0 14 76,55,2081
F5  1460 DATA 15,13,208,82 78 71
    82 65 77 32 76,69,78 71,84,
    72,1174
5D  1470 DATA 58,32,0,32,154,85,
    84,83,83,61,0 32 154,76,79 6
    7,1150
92  1480 DATA 75 83,46 13,13 0 1
    69,0,133,150,32 96,165 169,0
    ,141,1333
7E  1490 DATA 143 14,160,0,132,1
    24 105,0,2,240,17,200,174,14
    9,14,224 1784
BC  1500 DATA 16,240 9 232,142,1
    49,14,157,145,14,208,234,173
    ,149,14,96 1556
D0  1510 DATA 48,49,50,51,52,53
    54 55 56,57,48,49,50,51,52,5
    3 824
51  1520 DATA 54 32 60,225,169,0
    ,170,160,15,32,186,255,169,2
    ,162,53 1764
70  1530 DATA 160,15,32,180,255,
    32 182 255,32,231,255,169,8,
    160,0,170,2155
50  1540 DATA 32,180,235,173,140
    ,14,162 150,160,14,32,189,25
    5,32 182 255 2250
2F  1550 DATA 162 8,32,198,239,5
    ,32,14,15,00,127 133,00,32,1
    4,15,1828
1D  1560 DATA 160,165 00,152,32
    206,198 169,32,32,210,255,32
    ,14,15 240,1899
0E  1570 DATA 5 32,210,255,208,2
    46,169,13,32 210,255,208,213
    ,76,24,15 2171
A0  1580 DATA 32,207,255 170,169
    ,144,208,2,138,96 32,204,255
    ,185,0,170 2255
00  1590 DATA 32 180,255,32,231,
    255 104 104,169 45,160,27,32
    ,210,255 136 2242
7E  1600 DATA 208,250,169,13,76,
    210,255,73,48,134,00 132,09,
    160,0,177,2092
37  1610 DATA 00 240,6,32,210,25
```

```
    5,200,200,240,96,32,55,15,32
    ,107,14,1836
41  1620 DATA 173,145,14,240 46
    201,4,208,244,162,1,32,101,1
    5,72,32,1694
0E  1630 DATA 101,15 170,104,168
    ,24,95,32,116,15,220 10 96,1
    29,133,92,1529
0C  1640 DATA 32 116 15,5,92,96,
    169 145,14,232,201,50 144,2,
    233 7,1585
26  1650 DATA 41,15,96,162,174,1
    50,167,56,96,157,78 84,69,82
    ,32,210 1718
C7  1660 DATA 213,206,45,153,198
    210 50,32,36 0,157,78,84,69
    ,82,32 1731
E6  1670 DATA 204,207,215,45,205
    ,157 205,50 32,36,0,13,204,7
    3 78,75,1847
B3  1680 DATA 65,82,50,13,39,36,
    39 45,32,156,73,82,44,32,39
    33 812
0F  1690 DATA 39,45,155,82,85 78
    ,67 72,32,46,13,13,0,155 82
    85,1129
F2  1700 DATA 78 67 72 73,78,71,
    32 59,32,0 150,73,76 69 78 6
    5,1120
F8  1710 DATA 77,69,59,32,0,211,
    75,73,HO 32 36,0 76,52,3,163
    ,1037
43  1720 DATA 96,23,162,0 165,0,
    15,157,52,3 232,224,152,208,
    245,76 1003
70  1730 DATA 52 3,208,218,88,32
    ,63 245 163,13,76,210 255,21
    1,65 86 1335
30  1740 DATA 73,78,71,227,46,85
    ,218,13 0 162,70,160,14,32,5
    5,15 1330
47  1750 DATA 173,5,5,133,136 56
    ,173,6,5 233,16,133,137 165,
    136,32,1553
2F  1760 DATA 205 105,162,88,160
    14 32 55,15 160,1,165,137,2
    40,16,200 1039
F3  1770 DATA 24 165 136,233,253
    ,133 136 165 137 233 0,133,1
    37,200 236 152 2481
F3  1780 DATA 170 169 0 32,205,1
    05,162 56 160,14,76,55,15,13
    ,208 82,1646
0E  1790 DATA 79 71,82 65 77,32,
    76 69,78,71,84,72,50 32,0 32
    ,978
28  1800 DATA 194,85,84,69,83,01
    ,0 32,194,76,78,67,75 83,46
    13,1245
CC  1810 DATA 13,0,169,0,133,150
    ,32 90,165,169,0 141 148,14
    160,0,1435
74  1820 DATA 132,124,165 0,2,24
    0,17,200,174,148,14,224,16,2
    40,9,232,1550
C1  1830 DATA 142,145,14,197 145
    ,14 208 234,173,145,14 96,40
    ,48,50 51,1697
7A  1840 DATA 52,53,54,55,56,57,
    48,49,50 51,52,53,54,32,60,2
    29,1013
06  1850 DATA 169 8,170,160,15,3
    2 186,255,169,2,182,53 160,1
```

```
    5,32,169,1777
0D  1860 DATA 255,32,152,255,32,
    231,255,169,9 160,0,170,32,1
    88,255,173,2405
0E  1870 DATA 145 14,162 150,160
    ,14,32,189,255 32,152,255 16
    2,0,32,158,2004
B4  1880 DATA 255,5,32,14,15,96,
    127,133,88,32,14,15,169,166,
    88,152,1400
71  1890 DATA 32,205,189,168,32,
    32,210,255,32,14,15 240,5,32
    ,210,255,1927
46  1900 DATA 208,246,165,13,32,
    210,255,208,213,76,24,15,32,
    207,255,170,2333
42  1910 DATA 165,144,208,2,198
    96,32,204,255,169,0,170,32,1
    55,255 32,2105
D3  1920 DATA 231,255,104,104,16
    5,45,160,27,32,210,255,136,2
    00,250,169,13,2360
1D  1930 DATA 76,210,255,73,46,1
    34,88 132 89,160,0,177 88 24
    0,6,32,1008
09  1940 DATA 210,255,200,208,24
    6 56 32 55,15 32,107,14 173,
    149 14,240 2046
47  1950 DATA 46,201,4 208,244,1
    62 1,32,101 15 72,32,101,15,
    170,104 1508
02  1960 DATA 168,24 96,32,116,1
    5 220,10 96 128,133,82,32,11
    6,15 5 1307
23  1970 DATA 92,96,109,149,14,2
    32,201,50,144,2,233,7,41,15,
    96,162,1731
78  1980 DATA 174,160,167,56,96,
    187,78,84,69 82,32,210,213,2
    06,45,153,2062
53  1990 DATA 196 210,50 32,36,0
    ,187,78,84,69,82,32 204 207,
    215 45 1745
60  2000 DATA 205,157 206,50 32,
    36 0 13 204,73,78,75 69 82 5
    0,13 1350
80  2010 DATA 33,36 38 45,32 196
    73 82,44,32,39,33,39 45 155
    ,82 1051
0A  2020 DATA 85,78,67,72,32,46,
    13 13,0 155,82,85,78,67 72,7
    3,1058
FE  2030 DATA 78,71 32 58,32,0 1
    58,73,76,69 70 65 77,65,58,3
    2,1066
57  2040 DATA 0,211,75 73,90 32,
    36,0 76,52 3,163 48,162,0,10
    5,1200
A0  2050 DATA 0,16,157 52,3,232,
    224 152,208 245,76 52,3 0,5
    243,1712
C4  2060 DATA 234,0,201,0,0,0 0,
    0,169 0 133 250,169 144 133,
    251,1684
22  2070 DATA 169,1,133,174 133,
    153 169,9,133,175,133 154 16
    5,88 133,252,2257
25  2080 DATA 169 156 133,253,16
    0,0,177,250,145,174,230,250,
    208 2,230,251,2798
0F  2090 DATA 230,174 208,2,230,
    175,165,250,157 252 208 234,
    165,251 157 253,3151
10  2100 DATA 208,228 169 243 13
    3 187,169,156 133,188 165,5
    133,183,169,0 2477
34  2110 DATA 133,185,160,0,185,
    195,156,240,6,32,210,256,200
    ,208,245 32,2442
0F  2120 DATA 207,255 240,251,20
    1,48,240,4,201 58 48,230 41,
    15,133 186,2357
20  2130 DATA 76,234,245 147,17
    17,73,78 80 85 84,32,68,68 8
    6,73,1464
33  2140 DATA 67,69,32,78,85 77
    66,69 82 13,17,67,65 83,61,4
    9,980
0F  2150 DATA 32,47 32 60 73,83,
    75,61,32,56,32,75,82,32,57,5
    8,839
39  2160 DATA 45,32 0 67,78,77 8
    0 69 67,84,79,62,0 0,0 0,757
16  2170 DATA 0 0,0,255,255,255,
    255 255,0,0,0,0,0 0,0,0,1275
```

Super Index

Dig out those old magazines and get them organised with this super database

Have you ever spent fruitless hours sifting through a mountain of back-dated magazines, searching for a particular article, sub-routine or program? Now with Super Index and a little typing skill on your part, all this thumbing through pages can be a thing of the past

This database will enable you to select a subject from a cursor-driven menu and then tell you immediately the titles and page numbers where the subject is broached in your pile of magazines Press the 'R' key, and you are returned instantly to the subject list, ready to make another enquiry

This high speed shuttling between subjects and magazine titles is possible because the database is committed to memory only once before you actually use the program The system is designed this way so that the READ statement is not required to re-search the database or arrays The resultant speed means that information comes to you as fast as Basic can print it on the screen

The program begins by defining two user-defined variables The print limit (PL) is concerned with the size of the selection list displayed on the screen With PL set to 20 you get the maximum display of 20 items If you wish, this can be reduced

The next variable involves the alphabetical sort (AS), and is simply the memory location for the machine-code sort Though you may relocate the sort, you certainly can't omit it

The main task of determining the size and shape of the database is solved by reading all the data statements and concluding with values for DM (DiMension) and CL (CoLumn)

What may have caught your eye in the listing, is the isolated data statements that lie between the program and data information Lines 1760 to 1790 serve to format your screen displays

```
TL$(0)-1760 DATA C/MAGS 20MAY88     Title of the data base
TL$(1)-1770 DATA "MAG"              Headings for
TL$(2)-1780 DATA "PAGE"             the information lists
TL$(3)-1790 DATA *                  An end marker
```

What is so important about these particular data statements? Well, the entire usable database is structured on two parallel arrays The first is isolated and contains the various subjects on file and is known as the A$() array

The second, which is two dimensional, contains the information you're looking for and is known as the B$(-,-) array Its structure relies on the format defined by the above data statements (lines 1760 to 1790) The aim of this particular database is to locate the exact edition of the magazine plus the page number against any subject you have chosen from the subject list For example, an article entitled MODEMS can be found in the March 1988 edition of Your Commodore page 96 That precise information is written down as

```
1860 DATA MODEMS,YOUR COMMODORE MAR 88,96
          ↑                ↑          ↑
         A$()            B$(-,-)
```

The beauty of a system using a two dimensional array is its sheer flexibility, consider adding the following line

```
1765 DATA "VOLUME"
```

You could now go on to create an extra element in your database, and line 1860 could now look like this

```
1860 DATA MODEMS,4,YOUR
     COMMODORE MAR 88,96
```

So if you store your magazines in folders or volumes, you could easily grab Volume 4, in this case, whip out the mag and flick through to the correct page

If you really wish to impress your friends, you can use the program for other purposes Car performance figures could be yours at the touch of a button Try these lines

```
1760 DATA CAR PERFORMANCE           The title of your new database
1765 DATA "0-60"
1770 DATA "ECONOMY"                 Six statements to
1775 DATA "POWER"                   control six elements
1780 DATA "SERVICE"                 of the database
1785 DATA "MAX SPEED"
1790 DATA *                         DON'T FORGET THE END
                                    MARKER
```

by N Hart

And a typical example for your database would be

1860 DATA FORD FIESTA,
16 1 SECS,41 MPG,
40 BHP 6000 MILES,80 MPH

The only thing to remember is that your formatting statements (lines 1760 to 1790) must end with an asterisk (*), and that your database statements must contain the appropriate number of elements – a total of three in the case of Super Index, and six in the Car Performance example

With the database sized up and about to be committed to memory, it would seem appropriate to prepare a list directly for the screen, ie print A$() But first think about the disadvantages this may have Such a simple list would be chronological, in other words in the order that it's read, not alphabetical. Perhaps you would consider an alphabetical-sort to be a bit of a luxury, and I would agree if the database was small but, with anything larger than 80 to 100 items, luxury takes on a new meaning

The second problem would be the lack of a condensed list. It would be far better if the list routine examined itself for multiple entries and formed single entries in their place For example, modems are a very popular subject and occur three times in the database I've provided However, if 'Modems' is to be listed on the screen it need only appear once, any more would be unnecessary

The third problem is making a single item list seem intelligent. Each subject in the list needs to keep track of where it came from in the array and then it can instantly reveal the corresponding information Select MODEMS and you will be told that articles on this subject can be found in three separate magazines

How is all this achieved? The first trick is to reconfigure the A$() array so that each element includes its own subscript For example, the flavour of the month, MODEMS, occurs at positions 1, 20 and 30 in the array, and what you have at this point is

A$(1) = "MODEMS"
A$(20) = "MODEMS"
A$(30) = "MODEMS"

What you need is

A$(1) = "MODEMS1"
A$(20) = "MODEMS20"
A$(30) = "MODEMS30"

The method I've chosen involves printing to and reading off the screen Notice the OPEN 1,3 in line 330 To see what's involved, examine lines 540 to 640 and you will notice that I've formatted the printing on screen so that the variable (I) always begins four spaces from the end of each printed string, and that a colon( ) appears at the end of each string to minimise the string length to be read In this example I've used a "–" to illustrate spacings

	MODEMS
Changes to	MODEMS–20–
Which becomes	MODEMS–20–

All this can be observed as it actually happens by changing the value of POKE 53281 in line 330 to POKE 53281,11 (ie change the colour of the screen)

Now that the A$() array has been rewritten, a quick excursion through the land of machine code (SYS AS,A in line 680) is all that you need to return the array nicely sorted and ready for final processing before appearing on the screen. If you follow the simplified block diagram (Fig. 1) and read what is to follow, this last process will become clear

The overall task of the routine (lines 700 to 830) is to extract the stored information held by the A$() array and totally reconfigure and condense that information and present it as the L$() array (the list array) This takes place while the A$() array is progressively nullified in order to save memory The resulting L$() array eventually exists as an alternating series of words and numbers, hence the double dimension requirement expressed in line 470

The relationship between the L$() lines is very important Consider the following.

L$(29) = "MODEMS"
L$(30) = "1—20–30–"

Take the 1, 20 and 30 from L$(30) and apply them to the B$(– –) array

As you can see, L$(30) holds the key to the whereabouts of the corresponding information in the B$(–,–) array for L$(29) Each of the numbers held by L$ is designated three spaces in the string, so the carrying capacity is the maximum string length divided by three (approximately 85 locations) In our modems example you would need more than 85 articles entitled MODEMS before the program crashes with a 'string too long' error

If you've checked the program listing (lines 700 to 830) against the block diagram (Fig. 1), you will see that I've had to tell a few white lies in order to save space in the diagram Remember that the diagram serves only to show the essence of what is occurring when the program is running, the real details are in the program itself You'll also notice that my first step (at line 700) is to nullify A$(0) In fact, when the A$() array is sorted, A$(0) is kept out of the alphabetical order because it is normally regarded by the sort-routine as a reserved string (usually a title string identifying the array) As you can see, I've reserved my titles elsewhere in a separate array, TL$(), choosing to bypass A$(0) as part of my database The overall result does not suffer in any way and the procedures are easier to follow with the database titles separated off

At last you're on to the screen Lines 880 to 1020 deal with the subject list and, if you've paid attention, you'll realise that only the odd numbered subscripts are printed to the screen – L$(1), L$(3), L$(5) and so on The variable 'I' is the fundamental counter at this point and the mathematics involved is pretty straightforward, check lines 890 to 910 All the user has to do is to select the required page (SPACE or SHIFT/SPACE), press RETURN, move the arrow cursor alongside the appropriate subject, press RETURN again, and the information (the magazine titles and page numbers) are printed on the screen immediately

Why is this process so speedy? Well, if you've selected MODEMS, it is displayed as number 15 in the subject list, and if you take 15 and

B$(0 1) = "YOUR COMMODORE MAR 88" B$(1 1) = "96"
B$(0 20) = "YOUR COMMODORE APR 88" B$(1,20) = "39"
B$(0,30) = "YOUR COMMODORE DEC 87" B$(1,30) = "56"

multiply it by two you get 30. If you then take L$(30), you should get the string "1—20—30—". Take the three values (1,20, and 30 – the variable 'Z' in line 1090) and apply them to the B$(–,–) array and the process is complete. The value of 'Z' is obtained by taking a look at the numbers in the appropriate L$() string in jumps of three (P=P+3 in line 1070) and, in this way, three digit values can be catered for. Press the 'R' key and you are returned, without delay, to the subject list ready for another go. The business of darting from one screen to another uses no memory, so you can play around without fear of crashing the program.

Before you start key bashing, a few bits of information and advice. The program lines containing the colons only are purely decorative and are there to separate the various routines throughout the program. By contrast, the full stops in the DATA statements act as shorthand repeater statements, study the DATA statements carefully to understand what is going on. As to the capacity, I've tested the existing database to approximately 640 lines, without any problems – and that's more than ten items per monthly magazine for five years.

```
F3  10 REM ********************
F4  20 REM *** SUPER INDEX ***
E5  30 REM ***             ***
A0  40 REM *** WRITTEN  BY ***
01  50 REM ***             ***
26  60 REM ***    N. HART  ***
A7  70 REM ********************
0F  80 REM
FF  90 REM  *  28 MAY 1988  *
1A  100 REM ********************
00  110 REM
F7  120 REM
0D  130 REM
41  140 PL=28:   REM PRINT LIMIT

AC  150
58  160 A$=S$(88) REM ALPHABETICAL M/C SORT
90  170
8E  180 :
31  190 PRINTCH$(8)"POKE53281,15 POKE53280,11 OPEN1,3
82  200 PL=PL-2 IFPL>20THENPL=20

7E  210 PRINT"{BLACK,CLR,DOWN3}"SPC(3)"PLEASE WAIT"
ED  220 PRINTTAB(3)"{DOWN}THE INDEX IS BEING SORTED" L=2 PRINT"{HOME,DOWN5}":GOSUB1140
0C  230
CA  240
70  250 POKE53280,0 POKE53281,2:READX IFX<>$6THEN250
17  260 READTL$ PRINT"{RIGHT,UP5,WSON}"TL$ CL=CL+1
25  270 READTL$ IFTL$<>"*"THENCL=CL+1:GOTO270
99  280 DM=-1 CL=CL-2
A4  290 POKE53280,6 READA$ DM=DM+1 POKE53280,0:IFA$<>"END OF FILE]"THEN290
C0  300 DM=DM/(CL+2)+1
00  310
78  320
00  330 DIMA$(DM),B$(CL,DM),L$(2*DM),TL$(CL+2) RESTORE
AF  340
90  350 J=-1
D5  360 POKE53280,0 J=J+1:POKE53280,5 READX POKEA$+J,X IFX<>$6THEN360
00  370 READTL$(I) IFI<CL+2)THENI=I+1:GOTO370

C0  380 TL$=TL$(0)
50  390 I=1:E=0
55  400 POKE53280,7
11  410 $EADA$(I)
ED  420 PRINT"{HOME, DOWN, SPC39, UP}"
30  430 PSINT"{CD3}"A$(I)"{SPC4}{CLEFT5}"I"{CLEFT UP}"
EA  440 IFLEN(A$(I))>31THENPSINT"{BLACK,CLR,DOWN RIGHT}ST$ING IN DATA TOO LONG ERRORC{DOWN}" PRINIA$(I) STOP
10  450 READB$(E,I)
D2  460 IFB$(E,I)="" "THENB$(E,I)=B$(E,I-1)
4E  470 E=E+1 IFE>CLTHENE=0 GOTO450
D9  480 GOTO450
61  490 INPUT#1,A$(I)
34  500 POKE53280,0 IFLEFT$(A$(I),13)<>"{END OF FILE}"THENI=I+1 GOTO490
C5  510
38  520
94  530 POKE53280,0 PSINT"{CLR,BLACK,DOWN7,SIGHT2,RVSON}ALPHABETICAL SORT"
17  540 SYSA$,A POKE53280,3 I=1:X=0 PRINT"{CL$,C0,DOWN}"
12  550 :
D9  560 POKE53280,0 A$(X)=""
06  570 POKE53280,13 X=X+1
A7  580 IFLEFT$(A$(X),13)="{END OF FILE}"THEN710
16  590 IFLEFT$(A$(X),LEN(A$(X))-3)=LEFT$(A$(X+1),LEN(A$(X+1))-3)THEN660
D8  600 POKE53280,4
94  610 PAINTRIGHT$(A$(X),3)" {LEFT}";
17  620 PAINT"{HOME,DOWN}" L$(I)=LEFT$(A$(X),LEN(A$(X))-4)
10  630 IFPEEK(1144)=32THENINPUT#1,L$(I+1) GOTO690
01  640 INPUT#1,Y$(Y) IFY$(Y)=""THEN600
B4  650 Y=Y+1 GOTO640
E1  660 PSINTRIGHT$(A$(X),3)" {LEFT}"; P=P+1 IFP=12THENP=0 PRINT
92  670 GOTO560
34  680 FORZ=0TOY-1 L$(I+1)=L$(I+1)+Y$(Z) Y$(Z)="" NEXT
9A  690 I=I+2 Y=0 P=0 PSINT"{CLR,DOWN}" GOTO560
B4  700
35  710 I=1 POKE53280,14 PAINT"{CLR,BLACK,DOWN7 RIGHT2,RVSON}CLEASING BAR@AB@{CD}"
2F  720 PRINTFRE(0) POKE53280,15
26  730
09  740 PRINT"{CLR,BLACK,DOWN}" L=0
72  750 PSINTTAB(2)I/2+ 5 PAINTTAB(8)"{UP}  ";:IFI/2+ 5=NTHEN PRINT"{RVSON}";
AC  760 PAINTL$(I) :FPEEK(1144+(PL*40)+8)<>32THEN680
7F  770 IFL$(I+2)<>""THENI=I+2:L=L+1:GOTO750
92  780 L=L+1 PRINTTAB(10)"*END OF SUBJECT LIST*"
CD  790 PRINT"{HOME}" GOSUB1130
    L=L-1 GOTO810
D9  800 PRINT"{HOME}":GOSUB1130
9E  810 PSINT"{RIGHT, RVSON}{SHIFT}{SPACE}=PAGE{RVSOFF, SPC3, RVSON}{RETURN}=ENQUIS?"
00  820
81  830 POKE198,0 WAIT198,1 C=PEEK(631)
39  840 IFC=13THENLL=L GOSUB1380 PSINT GOTO860
28  850 IFC=32ANDL$(I+2)<>""THENI=I+2 GOTO740
8F  860 IFC=168ANDI<>2*L+ITHENI=I-(2*(L+PL+2)) GOTO740
58  870 GOTO830
51  880 P=-2 IFN=0THENI=(I)-(LL*2):GOTO740
47  890
00  900
A0  910
AC  920 PRINT"{CLR, DOWN2, RIGHT2, $USON}"L$(N*2-1)"{DOWN}" L=1
82  930 P=P+3
A8  940 IFP>LEN(L$(N*2))THENPRINTTAB(16)"{UP}  END ." P=1 GOTO1040
DC  950 Z=VAL(MID$(L$(N*2),P,3))

1A  960 PRINT"{UP} ,,FORTS=0TOCL:PRINTTAB(5)CHR$(18)
FA  970 IFPEEK(1069)<>32THENTS=CL NEXT PAINT"{UP}", L=L-1 GOTO1040
55  980 NEXT
B5  990 FORTS=1TOCL:PRINT"{UP}", NEXT PRINT"{RIGHT4}{LEFT4}"(P+2)/3"{UP}"
29  1000 L=L+1 PRINTTAB(5)TL$(0+1), PRINT@$(0,Z):IFD=CLTHEND=0 GOTO1020
```

```
14  1010 D=D+1:DDTO1000
4D  1020 PRINT L-L+1
0F  1030 DDTO930
1E  1040 GDSUB1130
0E  1050 PRINT"{BLACK,RIGHT,RVSO
    N}{5PACE}=PADE{RVSOFF,SPC15,
    RVSONJ{R}=RESTART"
12  1060 :
CE  1070 POKE198,0:WAIT198,1 C=P
    EEK(631)
E0  1080 IFC=32THENPRINT"{CLR,DO
    WN2}" P=P-3 DDTO920
46  1090 IFC=82THEN1=(1)-(LL*2)
    GDTO740
40  1100 GDTO1000
60  1110
5E  1120
17  1130 PRINT"{HOME}";
24  1140 PRINTIA0{20}"{RVSONJSUP
    ER INDEX'
F5  1150 PRINT"{RIGHT,RVSON,UP}"
    TLR
50  1160 PSINT"{SIGHT,50,CY36 SP
    }'
43  1170 FORX=0TOL:PRINT"{RIGHT,
    CNJ}"SPC(36)"{CNJ' NEXT
EA  1180 PRINT"{RIGHT,SL,CP36,50
    }"
24  1190 RETURN
0E  1200
94  1210 :
B0  1220 DATA32,115,0,133,97,169
    ,128,133,98 32,115,0,240,7,5
    ,128 133_58_32 115
01  1230 DATA0,1R5,47,133,99,165
    ,48,133,100,1R0,0,1R5,97,209
    ,99,208,7 200,1R5_50
5E  1240 DATA209,99 240,20,24,1R
    0,2 177 99,101,99,72,200,177
    ,99,101,100 133
B7  1250 DATA100 104,133,99,144,
    221,160,5,177,99,133 102,200
     177,99,133,101,200
E7  1260 DATA2,189,102,158,101,2
    4,165_99,1R5,7,133,99,1R5,10
    0,105,0,133,100,165,101
3C  1270 DATA200 2,158 102 190,1
    01,200,4,1R5,102,240 10,133,
    105 1R2,0,134,103,134
30  1280 DATA104,1R5,99,133,100,
    1R5,100 133 107,240 224_240,
    114,24,1R5,10R,105
R4  1290 DATA3,133,10R,1R5 107,1
    05,0,133,107,230,103,208,2,2
    30,104 1R0,2,177,10R
D9  1300 DATA153,109,0,138,16,24
    0,1R0,5 177_10R,153,109,0,13
    8,152,2,200 246 170
8D  1310 DATA5R,229,109,144,2,16
    6,105,160,255,232,200,202,20
    0,0,165,112,197,109
9D  1320 DATA144,10,176,34,177,1
    13 209,110,240,230,10,26,100
    ,2,105,112 0,145
FF  1330 DATA100,138,16,240,160_
    5 105,10R,0 145 100,130_192,
    2 208_246,169,0,133
5A  1340 DATA105 1R5,101,197,103
    ,200,152,165 102,197,104,208
    ,14R,1R5,105,240,130,98
71  1350
GF  13R0
GG  1370
F9  1380 PSINT"{RVSON,UP,SIGHTJM
    OVE CRSR{RVSOFF} {RVSONJ{R}-
    RESTART"
1G  1390 PRINT"{HOME}"SPC(20)"{B
    FC3 RVSONJ{5}=SAVE{DOWN}"
3D  1400 N=0 A=0 PSINT"{SIGHTJ}>
    "
D3  1410 POKE198,0:WAIT198,1 C=P
    EEK(631)
32  1420 IFC=82THENRETURN
50  1430 IFC=83THEN1550
0D  1440 IFA=LLANDC=17THENA=A-1
    PSINT"{UP2}"
G2  1450 IFA=BANDC=145THENA=A+1:
    PSINT
DD  1460 IFC=17THENA=A+1 PSINT"{
    RIGHT2,UP} {LEFT DOWN}>"
90  1470 IFC=145THENA=A-1 PRINT"
    {RIGHT2_UP} {LEFT_UP}>"
35  1480 IFC=13THENPSINT"{SIGHT3
    ,UP} , GDTO1500
DC  1490 DDTO1410
00  1500 INPUT#1,N
85  1510 RETURN
CF  1520
C5  1530
30  1540
07  1550 PRINT"{CLR,DOWN3}SAVE"C
    HR$(34)TL$CHR$(34)"{UP3}"
E0  1560 POKE631,13 POKE198 1 EN
    D
12  1570
09  1580
B6  1590
7C  1600
5A  1610
5C  1620 DATAC/MAGS 28MAY88
FA  1630 DATA"MAG  "
30  1640 DATA"PAGE "
25  1650 DATA*
B8  1660
B5  1670
AC  1680
9A  1690
50  1700
0E  1710
1F  1720 DATAMODEMS,YOUR COMMODO
    RE MAR 88 56
51  1730 DATASUPER MOUSE, ,59
C0  1740 DATABASIC COMPILER,COM
    DISK USER MAR/APR 1988,1R
11  1750 DATAPROFESSIONAL PROGRA
    MMING  ,48
45  1760 DATAMULTI COLOUR, ,14
RR  1770 DATABASIC+, ,20
C0  1780 DATATAPE ARCHIVE, ,21
F0  1790 DATALINK AND CRUNCH M/C
    ODE  ,22
FC  1800 DATAMONITOR, ,24
09  1810 DATADISK LIBRARIAN II,
    ,28
AC  1820 DATAPSDGRAM COMPRESSION
    , ,31
70  1830 DATAMONITOR,.,33
RF  1840 DATARYIIND INTO THE 651
    0,YOUR COMMODORE JAN 88,13
FE  1850 DATALIST PAUSE,YOUR COM
    MODORE APR 88 14
R7  1RR0 DATACARTRIDGE-FINAL CAR
    TRIDGE III, ,18
0D  1870 DATACARTRIDGE-ACTION RE
    PLAY, , ,
01  1880 DATAAUTO START,.,22
99  1890 DATACLEASING TEXT SCREE
    N, ,24
1D  1900 DATATABULATION ON SCREE
    N, ,30
AD  1910 DATAMODEMS, ,39
0E  1920 DATAMUSIC, ,46
-A0 1930 DATAMAKING GEOS WRITIOX
    , ,47
13  1940 DATAARTIFICIAL INTELLIG
    ENCE, ,55
22  1950 DATAMUSIC, ,R5
1F  1960 DATAEXTENDED BACKGROUND
    S, ,69
    1970 DATATAPE TURBO,,,
5E  1980 DATACP/M ASSEMBLY LANGU
    AGE,.,70
E5  1990 DATAMUSIC, ,75
DD  2000 DATAOLD ROUTINE, ,79
07  2010 DATAMODEMS,YOUR COMMODO
    RE DEC 87,56
G2  2020 DATA"END OF FILE",,,,
    ,,,,,,,,,,,,
```

Technical Information

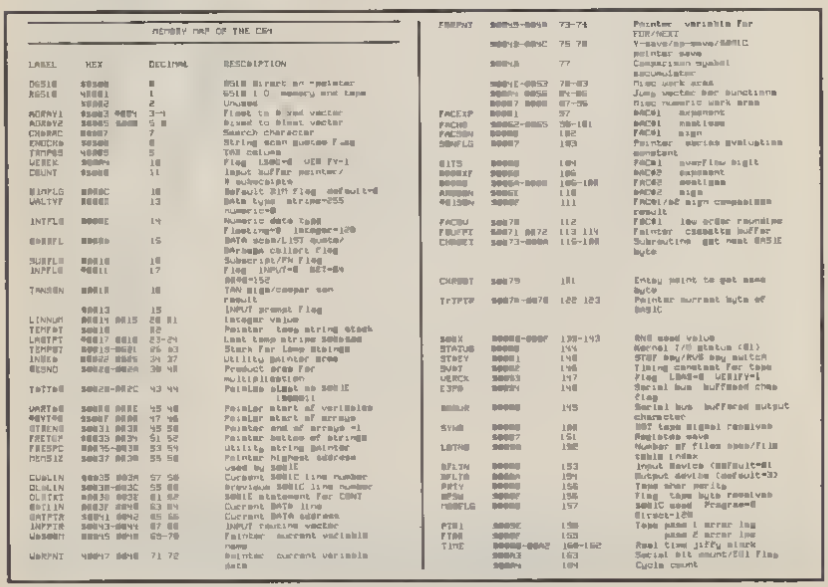
All you ever wanted to know about your Commodore but were afraid to ask.

Most programmers spend a lot of their time sifting through piles of technical books looking for the address of a certain routine or trying to find POKE to perform a certain function

Now you can throw away your books, as on the following pages you will find a wealth of information about all of the popular Commodore computers.

Advanced programmers will find the memory maps invaluable while both beginners and old hands alike will find the Hex converter, the hints and tips and much more to their liking.

Most of the information provided here is useful by itself Some information such as the addresses of routines within the ROMs, will be of more use when used together with a ROM disassembly



MEMORY MAP OF THE CBM

LABEL	HEX	DECIMAL	DESCRIPTION
D6510	$0000	0	6510 Direct Access Register
R6510	$0001	1	6510 I/O memory map type
	$0002	2	Unused
AORAY1	$0003-0004	3-4	Float to Fixed vector
AORBYZ	$0005-0006	5-6	Fixed to Float vector
CHARAC	$0007	7	Search Character
ENDCHR	$0008	8	String scan quotes flag
TRMPOS	$0009	9	TAB column
VERCK	$000A	10	Flag LOAD=0 VER F=1
COUNT	$000B	11	Input buffer pointer/ # subscripts
DIMFLG	$000C	12	Default DIM flag default=0
VALTYP	$000D	13	Data type strings=255 numeric=0
INTFLG	$000E	14	Numeric data type Floating=0 Integer=128
GARBFL	$000F	15	DATA scan/LIST quote/ GARbage collect flag
SUBFLG	$0010	16	Subscript/FN flag
INPFLG	$0011	17	Flag INPUT=0 GET=64 READ=152
TANSGN	$0012	18	TAN Sign/comparison result
	$0013	19	INPUT prompt flag
LINNUM	$0014 $0015	20 21	Integer value
TEMPPT	$0016	22	Pointer temp string stack
LASTPT	$0017 $0018	23-24	Last temp string $00=$00
TEMPST	$0019-$0021	25 33	Stack for temp strings
INDEX	$0022 $0024	34 37	Utility pointer area
DSSNO	$0026-$002A	38 42	Product area for multiplication
TXTTAB	$002B-$002C	43 44	Pointer to BASIC start of BASIC text area
VARTAB	$002D $002E	45 46	Pointer start of variables
ARYTAB	$002F $0030	47 48	Pointer start of arrays
STREND	$0031 $0032	49 50	Pointer end of arrays +1
FRETOP	$0033 $0034	51 52	Pointer bottom of strings
FRESPC	$0035-$0036	53 54	Utility string pointer
MEMSIZ	$0037 $0038	55 56	Pointer highest address used by BASIC
CURLIN	$0039 $003A	57 58	Current BASIC line number
OLDLIN	$003B-$003C	59 60	Previous BASIC line number
OLDTXT	$003D $003E	61 62	BASIC statement for CONT
DATLIN	$003F $0040	63 64	Current DATA line
DATPTR	$0041 $0042	65 66	Current DATA address
INPPTR	$0043-$0044	67 68	INPUT routine vector
VARNAM	$0045 $0046	69-70	Pointer current variable name
VARPNT	$0047 $0048	71 72	Pointer current variable data
FORPNT	$0049-$004A	73-74	Pointer variable for FOR/NEXT
	$004B-$004C	75 76	Y-save/op-save/BASIC pointer save
	$004D	77	Comparison symbol accumulator
	$004E-$0052	78-83	Misc work area
	$0053- $0055	84-86	Jump vector for functions
	$0056 $0060	87-96	Misc numeric work area
FACEXP	$0061	97	BASIC exponent
FACHO	$0062-$0065	98-101	BASIC mantissa
FACSGN	$0066	102	FAC#1 sign
SGNFLG	$0067	103	Pointer series evaluation constant
BITS	$0068	104	FAC#1 overflow digit
$0069	$006A	105	BASIC exponent
$006B	$006A-$006E	106-110	FAC#2 mantissa
ARISGN	$006F	111	BASIC sign comparison result
FACOV	$0070	112	FAC#1 low order rounding
FBUFPT	$0071 $0072	113 114	Pointer cassette buffer
CHRGET	$0073-$008A	115-138	Subroutine get next BASIC byte
CHRGOT	$0079	121	Entry point to get same byte
TXTPTR	$007A-$007B	122 123	Pointer current byte of BASIC
$008X	$0080-$009F	128-143	RND seed value
STATUS	$0090	144	Kernel I/O status (ST)
STKEY	$0091	145	STOP key/RVS key switch
SVXT	$0092	146	Timing constant for tape
VERCK	$0093	147	Flag LOAD=0 VERIFY=1
C3PO	$0094	148	Serial bus buffered char flag
BSOUR	$0095	149	Serial bus buffered output character
SYNO	$0096	150	EOT tape signal received
	$0097	151	Register save
LDTND	$0098	152	Number OF files open/File table index
DFLTN	$0099	153	Input device (default=0)
DFLTO	$009A	154	Output device (default=3)
PRTY	$009B	155	Tape char parity
DPSW	$009C	156	Flag tape byte received
MSGFLG	$009D	157	KERNAL used Program=0 Direct=128
PTR1	$009E	158	Tape pass 1 error log
PTR2	$009F	159	pass 2 error log
TIME	$00A0-$00A2	160-162	Real time jiffy clock
	$00A3	163	Serial bit count/EOI flag
	$00A4	164	Cycle count

CNTRN	00006	166	Tape sync countdown/bit count
BufANT	00000	100	Pointer Tape I/O buffer
INBIT	00007	167	RS232 input bits/temporary storage
BITCI	0098F	100	RS232 input bit count (after 1st)
RINONE	0099F	100	Flag RS232 start bit
RIBATP	00A00	170	RS232 input byte buffer / Tape leader count/parity
PIPNTY	00A0A	171	RS232 input parity
			Tape write leader/sense
SAL	00AAC 00AE	173 174	Pointer Tape buffer/screen memory
RAL	00AFE 00AF	174-175	Tape program end address
CMPE	00A50-00A1	176 177	Tape timing constants
TAPE1	00A92-00A3	178-179	Pointer start of tape buffer
BITT3	00A94	180	RS232 out bit count/tape leader count
NXTBIT	00A98	181	RS232 next bit to send/tape EOT
RINDATA	00A95	182	RS232 out byte buffer/read character error
FNLEN	00A97	183	Current filename length
LA	00A98	184	Current logical file number
SA	00A99	185	Current secondary address
FA	00A9A	186	Current device number
FNADR	00A9B 00AC	187-188	Pointer filename address
ROPRTY	00A10	190	RS232 out parity/input read input char
FBBLK	00A1E	190	Block left for tape read/write
MYCH	00A1F	191	Serial word buffer
CMDI	00AC8	130	Tape motor marker
STAL	00AC1 00CE	193-194	I/O start address
MEMUSE	00AC3 00CV	196-198	Kernal setup pointer/tape load address
LSTX	00AC5	197	Last key pressed
NDX	00AC6	198	Keyboard queue length
RVS	00AC7	199	Flag reverse chars on=1 off=0
INDX	00ACA	266	Pointer end of line for
LXSP	00AC5-00CA	001 200	Cursor row column at start of INPUT
SFDX	00ACF	203	Current key pressed no key=64
BLNSW	00ACC	204	Cursor blink phase on=1 off=0
BLNCT	00ACB	206	Cursor countdown timer
GDBLN	00ACE	000	Character at cursor position
BLNON	00ACF	207	Cursor blink phase on/off
GDBW	00AD0	208	Flag INPUT from screen/GET from keyboard
PNT	00AD1 0AD2	255 318	Pointer current screen line address
PNTR	00AD3	311	Cursor column on current line
QTSW	00AD4	212	Flag quote mode status 0n quotes=8 in quotes >8
LNMX	00AD5	213	Physical screen line length
TBLX	00AD6	214	Current row location of cursor
	00AD7	213	Last input/checksum/buffer temp data
INSRT	00AD8	218	Number of inserts outstanding
LDTBl	00AD9 00FE	217 242	Screen line link table
USER	00AF3 00F4	243-244	Pointer current screen colour RAM location
KEYTBL	00AF5-00F8	243-245	Keyboard decode table address
EIBUF	00AF7 00FB	247-248	Pointer RS232 input buffer
ROBUF	00AF3-00F9	245-288	Pointer 00000 output buffer
FBKZP	00AF8-00FE	251-254	Flag zero page area
BASPT	00AF7	355	BASIC Load save area
	00100-01FF	256 511	Processor stack
	00100-01PP	300-255	Float to ASCII work area
BAE	PP100-0132	000-319	Tape error log
BUF	00000-0259	313-000	System input buffer
LAT	00000-0259	001-01A	Logical File table
FAT	00A01-00EC	011-00F	File device number table
SAT	00A51-00TB	001-0170	Secondary address table
KEYB	00077 0280	131-040	Key0003 buffer
MEMSTR	00001 0020	041-018	Start of BASIC memory
MEMSIZ	PP103 0000	013-014	Top of BASIC memory
TIMOUT	00005	045	Serial bus time out Flag
COLOR	00006	018	Current character colour
SCRCL	00007	047	Background colour under cursor
AIEP3E	00008	048	Screen location open number
XMAX	30300	019	Size of keyboard buffer
RPTFLG	0020P	00F	Flag REPEAT key PIN default=8 repeat all=128 no repeats=64
KOUNT	00328	051	Repeat speed counter
DELAY	3b00C	032	Repeat delay counter
SHFLAG	00208	053	Flag SHIFT=1 CBM=2 CTRL=4
LSTNF	00001	004	Last shift pattern Flag
KSYLOG	00ZBF-00B0	055-055	Keyboard setup table pointer
MODE	00001	057	Flag 0=disable shift keys 128=enable shift
AUTOSW	00052	050	Scroll on=80 off=0

MSICTR	00293	659	RS232 control register image
MSICBR	00294	660	RS232 command register image
MSIAJB	00295 0296	661-662	RS232 non-standard baud rate
RSRTAT	00297	663	RS232 status register image
BITNUM	00000	664	RS232 bits left to send
BAUDBF	00298 0299	665-666	RS232 baud rate
RINBE	00000	667	RS232 index to end of input buffer
BIDDS	00295	695	RS232 page number of start of input buffer
00000	00000	660	RS232 page number of start of output buffer
00002	00300	670	RS232 index to end of output buffer
IRQTMP	00207-02A0	671 673	IRQ vector during tape move
ENABL	00291	673	RS232 enable/CIA6 INT1 interrupt control
	00292	674	CIA	timer A control reg during tape I/O
	00293	675	CIA	interrupt ipe tape read
	00294	676	CIA	timer B enable reg tape read
	00295	677	Screen line number
	00300	678	PAL/NTSC Flag 0=NTSC 1=PAL
	00203 A0FF	675-767	Unused
IRPRRR	00300-0301	768-756	Vector 0001C error messages (0E300)
INPAR	00302-0303	770-771	Vector 0001C warm start (0A963)
ICRNCH	00304-0305	772 773	Vector 0001C crunch tokens (0A57C)
IQPLOP	00306-0307	774-775	Vector 0001C print tokens (0A71A)
IGone	00308-0309	776 777	Vector 0001C start new line (0A7E4)
IEVAL	0030A-030B	778-775	Vector 0001C token evaluate (00BM1)
00000	0030C	756	Save accumulator
SIREG	0030H	781	Save X register
SYREG	0030E	782	Save Y register
SPREG	00307	783	Save status register
USRPOK	00310	784	USR function jump command (4C)
LRAMDR	00311-0312	785-786	USR address low/high form (0B248)
	00313	787	Unused
CINV	00314-0315	788 789	Vector Hardware IRQ (0EA31)
CBINV	00316-0317	790-791	Vector BRK interrupt (0FE66)
NMINV	00318-0319	792-796	Vector NMI (0FE47)
IOPEN	0031A-031B	794 795	Vector KERNAL OPEN (0F34A)
ICLOSE	0031C-031D	796 797	Vector KERNAL CLOSE (0F291)
ICHKIN	0031E-031F	798-799	Vector KERNAL CHKIN (0F20E)
ICKOUT	00320-0321	800-801	Vector KERNAL CHKOUT (0F250)
ICLRCH	00302-0323	802-803	Vector KERNAL CLRCHN (0F333)
IBASIN	00324-0326	804 805	Vector KERNAL CHRIN (0F157)
IBSOUT	00328-0327	800 807	Vector KERNAL CHROUT (0F1CA)
ISTOP	00329-032B	808 809	Vector KERNAL STOP (0F6ED)
IGETIN	0032A-032B	810-811	Vector KERNAL GETIN (0F13E)
ICLALL	0032C-032B	812-813	Vector KERNAL CLALL (0F32F)
USRCMD	0032C-032F	814-815	Vector USER VARS BLeL (0FE66)
ILOAD	00330-0331	816-817	Vector KERNAL LOAD (0F49B)
ISAVE	00332-0333	818-819	Vector KERNAL SAVE (0F5ED)
	00334 033C	820 827	Unused
TBUFFER	0033C-03FB	000 1013	Tape Header Buffer
	003FC-03FF	1000-1003	Unused
VICSCN	00400-0767	1004-0663	Screen RAM
	00768-07F7	0000-2032	Unused
	00778-07FF	2040-2047	Sprite Block data pointers (8 7)
	00000-0FFF	read 40058	BASIC RAM
	00000 0FFF	30798-40058	Eliminate ROM plug in case
	0A000-0FFF	40960 49151	Basic ROM/Alternate RAM
	0C000-CFFF	49152 53247	RAM memory
	00000 0002	53248 53329	VIC white registers (D000) /Character set
	00002F 03FF	53250-54271	Unused/Character set
	03400-0410	54272-54280	SID white 1800(1/Character set
	03418-04FF	54231-54527	Unused/Character set
	05000-07FF	54528-55000	SID images/Character set
	00000-00FF	53248-56319	Colour nybble memory /Character set
	00C00-0C0F	56320 56335	CIA 1 interface IRQ 165201 /Character set
	00C10 0CFF	56300 56575	Unused/Character set

E000-E0AF	57376 57519	Editor interface RAM	/Character set
E100-E7FF	57344 59391	Unused/Character set	
E800-FF00	57344 65280	KERNAL RAM/ RAM memory	
FF81 FFF5	65408-65525	KERNAL jump table ROM	memory

MEMORY MAP OF THE COMMODORE 128

ttBtt	HEX	DECIMAL	DESCRIPTION
D401D	D0000	0	6510 Direction register
R6510	D0001	1	6510 I/O memory and tape
RAMA	$0002 D00+	2-4	Jump mail for SYS
AREG	D0005	5	A register for SYS
XREG	D0006	6	X register for SYS
YREG	D0007	7	Y register for SYS
STATR	D0008	8	Search character
ENDCHR	D0009	10	String terminator flag
TMPMOS	D000B	11	TAB columns
VERCK	D000C	12	Flag 0=LOAD VERIFY=1
COUNT	D000D	13	Input buffer pointer
			# subscripts
DIMFLG	D000E	14	Default DIM flag DEFault=0
VALTYP	D000F	15	Data type 255=string 0=numeric=0
INTFLG	D0010	16	Numeric data type 128=integer=128
GARBFL	D0011	17	DATA scan/LIST quote/ Garbage collect flag
SUBFLG	D0012	18	Subscript/ FN flag
INPFLG	D0013	19	Flag INPUT=0 GET=64 READ=152
TANSGN	D0014	20	TAN sign/comparison result
CHKNKL	D0015	21	Current I/O channel
LINMUN	D0016 D017	22-23	Integer value
TEMPPT	D0018	24	Pointer temp string stack
LASTPT	D0019 D01C	25 28	Last temp string address
TEMPST	D001B D023	27 35	Stack for temp strings
TMPEX	D024+ D027	36 39	Utility pointer area
RESHO	D0028 D02C	40 44	Product area for multiplication
TXTTAB	D002B D02C	43 44	Pointer start of BASIC
VARTAB	D002F D030	47 48	Pointer start of variables
ARYTAB	D0031-0032	49 50	Pointer start of arrays
STREND	D0033-0034	51 52	Pointer end of arrays +1
FRETOP	D0035 D036	53-54	Pointer bottom of strings
FRESPC	D0037 D039	55-56	Utility string pointer
MEMSIZ	D0035-003A	57 58	Pointer top of RAM/ Highest address
CURLIN	0003B 003C	59-60	Current BASIC line number
TXTPTR	+003D 003E	61-62	Current byte of BASIC text
ENDPNT	D003F D040	63-64	Pointer item found by search
DATLIN	D0041 0042	65 66	Current DATA line
DATPTR	D0043 0044	67 68	Current DATA address
INPTTR	D0045 D046	69 70	INPUT routine vector
VARNAM	D0047-0048	71 72	Temper current variable name
VARPNT	D0049 D04A	73 74	Fa mier current variable data
FORPNT	D004B D04C	75 76	Pointer variable for FOR/NEXT
OPPTR	D005D 005E	77 78	@/actior table
OPMASK	D005+	79	Comparison symbol accumulator
DEFPNT	D005B D05!	80 81	Pointer current FN descriptor
DSCPNT	D005C-005+	82 84	Pointer current string descriptor
MEL?2B	D00555	00	Flag >2tb:t$T
JMPER	D0000 D007	00-07	0502 JMP to function
	D005B-0062	00-00	Temp data area for strings
FACEXH	D0053	00	FACE1 exponent
FACHO	D006+ 0007	100 103	FACE1 mantissa
FACSGN	D0008	104	FACE1 - sign
SGNFCB	D009C3	105	Pointer to/ps evaluator constant
			FACE2 exponent
A000XH	D009C1	100	FACE2 mantissa
A000M	D0003 D00A2	107 110	FACE2 sign
A000SN	D009C7	111	FACE1/FE2 sign comparison result
AH1000	D0078	112	
FACBU	D0077	113	FACE1 - low order rounding
FBUFAT	D0072 0073	114 115	Pointer - decimate buffer
AUTINE	D0074-0075	116 117	Increment value for AUTO
MBFCHP	D0076	118	Graphics mode set flag 1@=rol
WORE	D0077	119	Quotes flag/ store counter for LIST/RE
MULP	D0078	120	Counter
SYNTMP	D0079	121	Temp for character index
SSBESC	D007C D07C	122 124	Pointer DOS descriptor
T00	D007B-D07E	125 126	Pointer top of run-time stack
BLANKOD	D0077	127	Temp program direct mode
PARBTS	D0080	100	Disk command syntax check
PARPTR	D0081	129	Disk command syntax check
BLKOTX	D0082	130	
EDLSEC	D0083	131	Current colour
MULTI1	D0084	132	Multicolour 1
MULTI2	D0085	133	Multicolour 2
FORSHH	D0086	134	@/t key foreground colour
SCALEX	D0087 D088	135-136	Scale factor X
SCALTY	D0089 D08A	137 138	Scale factor Y

[illegible]	D008B		Flag Step ctlst
[illegible]	D008C D09C	138-143	Temp data area
[illegible]	D008D	144	KernalT I/O status (ST)
SIMKY	D008E	145	STOP key/RVS key switch
SAVI	D008F	146	Timing constant for tape
VERCK	D0090	147	Flag LOAD=0 VERIFY=1
C3P0	D0091	148	Flag Serial bus Salm Flag
D00UP	D0092	149	Serial bus shiftbuffer for output
SYNO	D0093	150	EOT tape signal received
	D0094	151	Register save
tBTNO	D0095	152	Number of files open/File table index
DFltN	D0096	153	Input device (default=0)
DFltO	D0097	154	Output device (Default=3)
tBTY	D0098	155	Tape char parity
DFSW	D0099	156	FTAG tape byte received/s
MMRIO	D009B	157	BASIC mode tape=+0 Direct=128
tTBI	D009C	158	Tape pass 1 error log
PTR2	D009D	159	pass 2 error log
TIME	D009E D0A2	160 100	Real-time jiffy clock
D0B6	D00E1	183	Secist Bit count/EOI flag
BUSAI	D00E2	184	Cycle count
CNTDN	D00A5	185	Tape sync countdown/Bit count
BUFPNT	D00A6	186	Pointer tape I/O buffer
INBIT	D00E7	187	BASIC input bit/tape write (er/read count)
BITCI	D00E8	188	BASIC input bit count/ tape write# (Err/EOF)
RTBYTE	D00E9	189	Flag RS232 start bit
RTBUTA	D00AA	170	RS232 input byte buffer/ tape (temp/counter/id)
CIPRTY	D00EB	171	RS232 input parity/ tape (write (er length/read checksum)
SALH	D00AC D0AD	172 173	Pointer tape buffer/screen scrolling
EALH	D00AE D0AF	174 175	Tape program end address
CMPB	D0000-0001	176 177	Tape timing constants
TCPEI	D00B0-00B3	178 179	Pointer start of tape buffer
BITTS	D00B4	180	RS232 out bit count/tape timer enable-1
NXTBIT	D00B5	181	RS232 next bit to send/tape EOT
BYTEIA	D00B6	182	RS232 out byte buffer/read character error
FNLEN	D00B7	183	Current filename length
LA	D00B8	184	Current logical file number
SA	D00B9	185	Current secondary address
FA	D00BA	186	Current device number
FNADR	D00BB-00BC	187 188	Pointer to filename
PSPEED	D00BE	189	RS232 out parity/Baud rate input char
FSBLK	D00BF	190	Blocks left for tape read/write
MYCH	D00C0	191	Serial word buffer
CMD1	D00CE	192	Tape motor sense
STALN	D00C1-00C2	193 194	I/O start address
MEMUSS	D00C3 D0C4	195 196	Kernal setup pointer/tape temp address
SDTA	D00C5	197	Tape read/write data
SA	D00C5	198	Bank for LOAD/SAVE/VERIFY
FNBANK	D00C7	199	Bank holding Filename (FNBANK)
DIBUF	D00CE-00C5	200-201	Pointer D0E3B Input buffer
RDOOS	D00CA-00CB	001 D46	Pointer RS232 output buffer
KEYTOB	D00CC D0000	204-205	Pointer key/board table
INPARM	D00CE D00E7	D46 007	Pointer Bitq for Kernal PRIMM
NDX	D00EB	046	Index to keyboard buffer
KYNRK	D00E1	209	Flag Function key=ness
KEYIDX	D00AF	210	Index to function key string
SHFLG	D00A5	211	Flag SHIFT=1 CBM=2 CTRL=4
SFBX	D00A5	212	Current key pressed @=row key
LSTX	D00A5	213	Last key pressed
CIRN	D00E0	214	Flag INPUT from screen or GET from keyboard
MODE	D00E7	215	Flag 40/80 columns 0=40 columns
GRAPHM	D00E5	210	Current GRAPHIC Mode
CHAMEM	D00E5	217	Flag VIC Fetch from ROM/RAM
PNT	D00E8-00EF	210-223	Programmable key variables
USER	D00EZ D0E3	220-227	Pointer current screen color RAM location
SCTOP	D00E6	228	Top line of window
SCBOT	D00E5	229	Bottom line of window
SCLF	D00E0	230	Left row of window
SCRI	D00E7	231	Right row of window
LSXP	D00E8	232	INPUT start column
LSTP	D00E5	233	INPUT start line
INDX	D00EA	234	INPUT End line
TBLX	D00EB	235	Current row
PNTR	D00E8	230	Cursor column
TINES	D00A5	237	Maximum number of screen lines
COLUMN	D00E2	238	Maximum number of screen columns
GDTAX	D00EF	239	Current character for printing
LSTCHR	D00FB	240	Previous character printed (ESC test)
COLOR	D00E6	241	Current character colour

```
TCOLOR   00AF2      242          SaVes character colour for
                                 INKT/DEL
RVS      00AA3      243          Flag  RVS characters
                                 0=off 1=on
STBW     00AF4      244          Flag  1=quotes 00AA on
                                 0=dell 00AA
INSRT    00AF5      245          number of inserts

INSFLG   00AF6      246          Flag  Auto-insert 00AA
LOCKS    00AF7      247          Flag  SHIFT or CBM* pressed
SCROLL   00AF8      248          Screen scroll disable
                                 0=enable
DEVICE   00AF9      249          CTRL # disable
FSTXEP   00AFA-00AA 250 E55      From zero page area
FBUFPTR  00B00-00BF 250 271      Filename manipulation area
XCNT     00B1B      E7E          00A loop counter
LDNFIL   00B11      273          Length of 00A filename I
DEVNUM   00B1E      27A          First drive number
LDNFIA   00B13-0B14 275-E7B      Address of 00A Filename I
LDNFIL   00B1E      277          Length of 00A Filename E
DEVNUM   00B18      278          Used=0 drive number
LDNFIA   00B17-0B18 278-00B      Address of 00A Filename E
DOSBAL   00B1A-01A  E81-00A      Start address for
                                 DOSBAS/DOSWA
DOSBFH   00B1B-01C  283 284      End Address for DOSWA
DOSLA    00B1D      286          00A logical File number
DOSFA    00B1E      288          00A device number
DOSSA    00B1F      287          00A Secondary address
DOSRCL   00B20      250          DOS record length
DOSBNK   00B21      290          00A bank number
DOSBIB   00B04-0B23 250-251      00A iobuffer
SIRCHK   00B24      292          DOS bit Flag
BAB      00B25      253          Pointer  USING begin number
END      00B26      294          Pointer  USING end number
DOLR     00B27      296          Flag  USING dollar
FLAG     00B28      B9B          Flag  USING comma
BWB      00B29      297          USING counter
USBN     00B3A      256          Sign exponent
UEXP     00B2B      250          Pointer  exponent
UN       00B2C      BBB          number of digits before
                                 decimal point
CHBN     00B2D      3B1          Using justify Flag
UF       00B21      3B2          number of Field characters
                                 before decimal point
NF       00B2F      3B3          number of Field decimal
                                 places
FBUF     00B32      3B4          Flag  +/- in USING Field
FISA     00B31      3B5          Flag  USING exponent
ITOF     00B32      3B6          Switch
CFORA    00B33      3B7          Field character counter
EXB      00B34      3B8          Sign number
BLFD     00B35      3B9          Flag  Blank or asterisk
BEBFO    00B36      31A          Pointer  beginning of Field
LFNR     00B37      311          Length of Field
ENDFS    00B3B      312          Pointer  end Of Ascii
BACK     00B3B 01AF 313 511      System stack
BUF      00200-0258 512-600      System input buffer for
                                 BASIC and MONITOR

FETCH    00AAZ      674          Subroutine  LDAC  ) Y from
                                 any bank
STASH    00AAF      687          Subroutine  STAC  ) Y to
                                 any bank
CMPARE   00AB2      702          Subrout ne CMPC  ) Y in
                                 any bank
JSRFAR   00AC2      717          JSR in any bank
JMPFAR   00AE3      739          JMP to any bank
ICRNCH   0038C-038D 78B-781      Vector  BASIC crunch tokens
UNTLOF   0B3BE 038F 782 783      Vector  LIST
IEVAL    00318 031I 70A 70B      Vector  evaluate mode
IKONE    00311 0313 700 707      Vector  BASIC character
                                 dispatch
IISO     00314-0316 708-70B      Vector  Hardware IRQ
IBRK     0031B-0317 750 751      Vector  BRK interrupt
INMI     00318 031E 75A-75D      Vector  NMI
IOPEN    0031A-031B 79A-79B      Vector  KERNAL OPEN
ICLOSE   0031C 031B 75B-797      Vector  KERNAL CLOSE
ICHKIN   00311 031F 79B-79B      Vector  KERNAL CHKIN
ICKOUT   00320-0321 9AA 9A1      Vector  KERNAL CKOUT
ICLRCH   0032E-03E3 A0A A03      Vector  KERNAL CLRCHN
IBASIN   00324-0325 A9A-9AA      Vector  KERNAL CHRIN
IBSOUT   00326-0327 9AA-9B7      Vector  KERNAL CHROUT
ISTOP    00328-032B 9AA-9AA      Vector  KERNAL STOP
IGETIN   0032A 032B A1B-A1I      Vector  KERNAL GETIN
ICLALL   0032C 032B A1E A13      Vector  KERNAL CLALL
EXMON    0032E 032F A14 A15      Vector  indirect monitor
                                 commands

ILOAD    00330-0331  A1B A17     Vector  KERNAL LOAD
ISAVE    0033K-0333  A1A-A1B     Vector  KERNAL SAVE
CTLVEC   00334-0336  9E0-9E1     Vector  CTRL 00AA link
SHFVEC   00338-0337  9E2-9E3     Vector  SHIFT mode link
CBMVEC   00338 0338  9E4-9E5     Vector  C=  mode link
KEYVEC   0033A 033B  E5B-987     Vector  keyscan indirect)
KEYCHK   0033C 0335  9E8-9AB     Vector  illeg keypress
DECODE   0033E-033F  A3B-83I     Vector  keyboard decode
                                 tables

         00340-0341  03E-041
KEYB     0034A-0353  042-051     Keyboard buffer
TABPAF   00354 0358  052-05I     81 temp 16 bytes
BITABL   0035E 035I  90A-9AA     Screen line link table
LAT      0036E-0B0B  0B5-077     Logical file table
FAT      0036C 036B  076-077     00AA line number table
EAT      0036E-037F  67B-09A     Secondary address 00AA
CHRGET   00380-038E  90A 926     Subroutine  get next 00AA
                                 byte
CHRGOT   00B0B 00AE  E62-52E

INCBBI   00307-03AA  527 03E     Subroutine  fetch into
                                 RAM 0
LMEMBE   00BA0-03BB  530-550     Subroutine  fetch into
                                 RAM 1
INDINI   00307 0387  BB1 569     Subroutine  Fetch INDEXI
                                 indirect

IBBIBB   00BC0-03CB  90A-90A     Subroutine  Fetch INDEXE
                                 indirect
INDTXT   00BC3-0381  90A 977     Subroutine  Fetch TXTPTR
                                 indirect

EEBB     00382-0384  978-980     Floating point constant
                                 From 90A
CURRA    9038E       BB1         Bank for PEEK/POKE/SYS
TMPBES   90388       90A         Temp area for INSTR
FIBBAB   90B3A       90A         Bank for string-number
                                 conversion
BABAIE   90B3B-03BC  907-90B     Temp area for BSHAPE
BITS     90B3F       BB1         FROM overflow Digit
BMPTMP   90B2B-03F7  90A 1B2B    Temp area for BMPBAU
VICSCN   90A00-07E7  1B24 3B23   40 column screen memory
SPRPTR   90F70 07FF  20B4-90A7   Sprite pointers
BLKBTK   90A00 09FA  2048 9000   BASIC bounds block
SWRCT    90A00 0BBI  2560 2661   Vectors external system
BEJMAU   90A2B       90BA        90AA/CBM STAT status
PALNTS   90B03       2563        Also  PAL/NTSC
INIIBT   90BA4       2564        Flag  Reset vs NMI for
                                 initialization
9EABIA   90A05-90AA  2B6B-2B6B   Bottom of systems bank
                                 memory
MEMSIZ   90A07 90BA  2567-2568   Top of system Ram memory
IBBTEMP  90A2B 90BA  2568-2570   Temp store for IRQ vector
                                 During Stop I/O
CASIBM   90A6B       2571        TBB comes during Tape 90A
STUPID   90A2B-0AB0  2572 2573   Tape read temp
TIMOUT   90A3E       2574        Serial bus time out Flag
BMKBL    90A2B       2575        RS232 enables (ACK)
                                 (all=trunk control
MSICTR   90A10       2576        RS232 Receiver register
                                 image
MSICBR   90A11       2577        RS232 transmit register
                                 image
MSIAUB   90A1E-0A13  2578 257E   RS232 non-standard baud
                                 rate
MSISTAT  90A14       2B8B        RS232 status register
                                 image
BITNUM   90A15       2581        RS232 bits left to sent
BAUDOF   90A16-0A17  3B8E 2583   RS232 baud rate
RIDBI    90A1B       2584        RS232 index in end Of input
                                 buffer
RIDBA    90A19       2585        RS232 save number of start
                                 of input buffer
ROBAB    90A1A       2586        RS232 temp number of start
                                 Of output buffer
ROBAB    90A1B       2587        RS232 index to end of
                                 output buffer
SERIAL   90A1C       2588        Flag  last serial
                                 internal/external op
TIMER    90A1B-0A1A  2589 2591   Incrementing jiffy register
BSNK     90A2B       2592        Size of keyboard buffer
PAUSE    90A21       2593        Flag  CTRL S
BFIFLG   90A22       2594        Repeat key flag  default=0
                                 repeat all=25
                                 no repeat=6B
REPNT    90A23       2595        Repeat speed counter
DELAY    90A24       90AB        Repeat ^ 4y counter
LSTBMF   90A25       2597        Last MATRIX value Flag
BLNBN    90A26       2598        Flag  VIC cursor Blink
BLNBN    90A27       90AB        VIC cursor blink enable
BLNCT    90A2B       90AB        VIC cursor blink timer
GBBLN    90A29       90A1        VIC character under cursor
GBCBC    90A3A       90AA        VIC background colour under
                                 cursor
CURPBA   90A2B       2603        VBC active cursor addr
LNV      90A2B       2564        VIC tmi screen start 00AA
VMC      90AA0       90A5        VIC Pil map start page
LNTS     90A2B       90AB        VBC last screen base
LPX      90A2F       2607        VBC sprite max x
LINTMP   90A0B       90AB        Temp pointer For LOOF
SAVBB    90A3|-0A3V  2609-B1E    Temp data For VBC screen
                                 Handling
CURCBL   90A3B       BB|3        VBC colour under cursor
SPLIT    90A3B       BB14        VIC split screen raster
                                 value
PAUSEV   90A37       2615        1 register save For pork
                                 scan
PALCHI   90A3B       2B10        Jiffy adjustment For PAL
                                 system
XCNI     90A3B-0A3F  9B0B-2713   PLA bankers buffer
         90A4B-0A4A  2720 B730   ALM temp data
LBMBTH   90A4B       B731        Flag  Load/B/Kissassemble
         90A2F-0AB1  E732-E737   Temp PLA values
XSAV     90A0B       E75B        X saves during indirect
                                 subroutine calls
BIRCTN   90A03       E739        Direction indicator For
                                 tapesoFer
TBFPS    90A01 0A3F  E7A8 27E1   PLA temps
CURBANK  90AC8       2752        Function key BOM bank being
                                 called
PAT      90AC1-0A7F  2753-2B15   Table of toggle ROM cartbs
TBUFFB   90A3B 90AF  9B1E-3B07   Tape buffer
         90ABB-0BFF  3B0B 3B71   Blnk 00BT page
90A3B1   90CB0 0CFF  3B72-33B7   00B32 input buffer
90A3E0   90B00-0B7F  3320 3583   00B32 output buffer
         00B3E 0FF   3B0A 43B4   From 00BBB
PKTBUF   01000-1000  90A0 9100   Function key storing lengths
                                 table
PKYBUF   01B00-1BF7  9100-4351   Function key definition
                                 table
         01100 1100  4352-4B00   CP/M reset subroutine
XPBN     01131 113F  4901-4907   Current pixel X position
YPBN     01133-1134  4903 4904   Current pixel Y position
IOCST    01135 113S  4905 4906   X co-ordinate destination
YOCST    01137 113B  4907-4908   Y co-ordinate destination
IXPA     01139 113A  4909-4910   X position For BASIC
IYPA     01132 113C  4911-4912   Y position For BASIC
XPSN     01130-113E  4913-4914   X parameter sign
YPSN     01130 1140  4915-4916   Y parameter sign
```

	$11W7	T7W9	%W17 W%29	time drawing temps
ZBRWAL	$1TW5	T7W8	%W21 %W2E	Graphics error veTue
LTGBER	$1197		%W23	Graphics Tenmes marker
SBFRT8	$1198		%W2T	Graphics greater marker
ANDBMN	$119B		%W29	Sign of angTs
SIAWAT	$119A	T79B	%W28 %W2T	Sin value of angIs
COSWAL	$19C	T79E	%W2E %W3E	Cosine value of angle
ANNCNT	$11AE	T7W7	%W3B %W3T	Tmage Ter angle-systemes FeaTTmee
XCIRCT	$1TSE	T1ST	%W3E %W33	CIRCLE centre X pos/BBX point T X
YCIRCT	$1TSE	T1S3	%W34-%W35	CIRCLE centre Y pos/BBX point 1 Y
STRSZ	$1153		%W36	Shape string Tength
XRADUS	$119A T1S5		%W38 %W37	CIRCLE X reeiue/BBX rotation angTs
STTYP	$119A		%W38	RegTaes shape mode
STRT8	$119B		%W37	String position counter
YRADUS	$1T5E T157		%W38-%W3B	CIRCIT 7 radius
NEWBYT	$1T87		%W3B	STS bit mes Byte
SCTBWS	$1T88	T1S9	%W3B-%W4T	New string or bit mag byte
XBITBR	$1T89	T1SA	%W41-%W42	Circle rotation angTs
SBRLEN	$1T8A	T1SB	%W42-%W43	Shape seTuen Tength
YBLEE	$1T8B	T1SC	%W43 %W44	SEX Tength of a qiee
ANSBEB	$1T8C	T1SD	%W45-%W49	Acs angTe 4Isrt
ANSBNB	$1T8T	T1S1	%W48 %W47	Ans angTs end
STBABS	$1187	T1W8	%W47 %W48	Save shape string beacrister
XRCOS	$1T88	T191	%W49 %W4B	X radius * COS(angTs)
SITIBX	$1T8T		%W4T	Bit index inte byte
YRSIN	$1T8B	T193	%W4B %W4I	Y reeiue * SIN(angle)
XRSIN	$1T8A	T195	%W52 %W53	X reeiue * SIN(angTs)
YRCOS	$1T8B	T197	%W54 %W52	Y radius * COS(angTs)
CHRPAS	$1T89		%W56	Sigh byte of character ROM address
ZTICNT	$1T8D		%W57	Tmap for BBMAPE
SCATEN	$1T8E		%W5B	TTag enaTe emeds
WTSTH	$1T63		%W59	Fing doubTe eidth
TYL7L8	$1T8C		%W6B	Fing FiTT bee
SITMSK	$1T6A-T192		%W6I %W6B	Temp Ter bitmask
TRCPT8	$1T9T		%W63	0-trans off 255-trans on
RSWAB8	$1T7B-T173		%W64 %W67	Temps for RENAPHEE
WTEMP	$1T74-T178		%W6B %W7B	Graphics temp storBeg
ASBMWT	$1T7A T173		%W7A-%W7B	Flag convert fimaiipa pernt to integer
ADBAYZ	$117E T192		%W7B-%W7T	Tipa nonvect integre to Timatipa point
SDATA	$1191 T1B5		%W7B %W85	Spcite meeds and Siraction tabie
VICSAV	$11B9 T1FT		%W86 %W87	Copy of VIC registers
DLBLIN	$1200 T201		%W88-%W8B	Previous BASIC line
OLDTXT	$1202-1203		%W1B-%W1T	BASIC statement for CONT
PUFILL	$1204		%W12	Fill symbol for UM1MB
PUCMMA	$1205		%W13	Comma symbol for UM180
PUDOT	$1206		%W14	DecimaT point symbol for UB1MA
PUMONY	$1207		%W15	Dotter/pound symbol for UBTMB
TMPNUM	$1208		%W16	Last error number
TMPLTN	$1209 T20A		%W17-%W1B	Last error line number (BBBBSMemory)
TMPBND	$120B T20C		%W1B-%W2B	Line number for TMAP (ZBS-eFF)
TMPTBP	$120B-120T		%W2T %W23	Temp for TBAP number
TXTTBT	$120D T21T		%W24 %W25	Pointer top of BASIC text
MXMEMB	$1212 T213		%W26-%W27	Pointer top of bank B storage
TMPTXT	$1214-1217		%W2B-%W31	BB/LBBT temp
USRPOK	$1219 T21A		%W32-%W34	USR vector base
RNDX	$121B T21E		%W35-%W3B	BAS base value
CIBCTE	$1220		%W3A	Degrees per circle segment
STJAVU	$1221		%W41	CeTd/swFe reeet status
TEMPO	$1222		%W42	Tpear rate
WOITSB	$1223 T228		%W43 %W4B	
NTTME	$122B-122A		%W4B-%W5B	
OCTAVE	$122C		%W5C	
PITCH	$122B-122T		%W53 %W54	
VOICE	$122T		%W5B	
WAVES	$1289 T230		%W5B %W5B	
SNBIT	$1283		%W6B	
TLTBA	$1284 1237		%W6B-%W63	
7TTTL8	$1238		%W6A	
NIBBLT	$1239		%W6B	Temp Ster Ter ENAVLBPE parameters
TONNUM	$123A		%W6C	Current ENAVELOPE number
TONWAI	$123B T23B		%W6T %W6B	Current ADSR and waveform
MANCNT	$123T		%W7B	CounTer Ter envaTeep parameters
ATKTAB	$123T T1YB		%W7T-%W8B	ENAVELOPE attack/decay tabTe
SUSTAB	$1Y4B T282		%W8T %W9B	ENAVELOPE maintain/reTease tabie
WAVTAB	$1B5E T25C		%W9T %W7B	ENAITBPE waveform table
PULSTB	$1B7B-125B		%W7T %W1B	PuTee eidth Tow byte tBBle
PUTBHI	$1B9T T27B		%W1T %W2T	Pulse width high byte tabTe
TTITBS	$1C7T 1T7S		%W2T %W2B	TiTter vaiume tabie
TPPTIB	$1Y7B-T27B		%WEB %W2B	FTags interrupt handTing trigBeS
S01ATT	$1Y7B		%W8B	Iina Ter speTte-speiis enaTimien TMB beacTTng (Tow)
S01WTT	$1Y7A		%W3B	Iina Ter speTte-Bata enaTimien TMB handTTng (Tow)
SPTWTE	$1Y7B		%W2T	Iina Ter Tightpen TBB handTing (Tow)
S01WTH	$1Y7C		%W3E	Iina Ter spriie-speiis TBB ihii
S01WTH	$1Y7D		%W33	Iina Ter spriie-Bata IBB ihii
SPINTH	$1Y7E		%W3H	Iina Ter Tightpen TBB ihii
INTVAL	$1Y7F		%W3S	Fips caiiaiaon enaTpa

COLTYP	$1Y9B		%W3B	Coiiision Interrupt type
VOICE	$1Y8T		%W3T	Voice number for SBUND
TTWELB	$1Y8E-1Y9A		%W3B-%W4B	SBUND time Tow byte
TTHIGH	$1Y9B-1Y8T		%W4T %W5B	SBUND time hT byte
HWALB	$1Y9A-1Y9B		%W4A-%W4B	SBUND
MWANT	$1Y9C	T9A5	%W4T-%W4B	SBUND
HTWLB	$1Y9D	T9A7	%W4B-%W5T	SBUND
HIMBI	$1Y9T-1Y83		%W53-%W5B	SBUND
STRCTW	$1Y9A	T9A6	%W4B-%W5B	SBUND direction taBTe
STEPLB	$1Y9T	T9A9	%W5B %W61	SBUND step vaTues Tow byte tBBTe
STEPHI	$1Y9A	T9AC	%W6T-%W6A	SBUND step vaTues hT byte tabTe
FREQLB	$1Y9B	T9AF	%W6E %W7T	SBUND Treqaency values Te-byte taBTe
TRBENT	$1Y9C	T9A2	%W6T-%W7B	SBUND Treqaency vaTues Hi-byte taBTe
TTNE	$1AA3	T9A4	%W7T %W7B	Duration Ter SBUND
	$1Y9C	T9A8	%W7A-%W8A	Tmpa Ter SBUND
PBTTMP	$1Y8A-1Y8B		%W8B %W8A	Temp store Ter TightpeN
	$1Y8T	T9FF	%W8T-%W83	SBTRBAA/BTRBET staTpas

COMMODORE 128 MEMORY OVERVIEW

HEX	DECIMAL	DESCRIPTION
$0000 T2FF	0 B453	BASIC workspace
$1C00 A4AB	T838%-41B46	BASIC ROM
$4000 AFAB	16384-19B46	Empty ROM space
$A000 AFFT	%W1B-%W2B	BASIC kernel ROM
$B000 AFAT	%W2C-%W4T	BASIC jump tabTe
$AF00-AFFF	%W5B-%W5B	Empty ROM space
$D000 DFFT	%W6B-%W1T	VIC/TTB
$D000-DFFF	%W1C-%W2T	Screen/keyboard routines
$D000-DBCC	%W2B-%W3A	VIC chip rom CBNT
$D0CT	%W3B	128 base extra keyBoars Tines (CBNTTNL)
$D03B	%W3E	12% ends system cTock symbs register
$D12B-D11C	%W72-%W3B	SID chip rom CBNT
$D500	%W5B	MMU primary configuration register
$D501	%W5B	MMU Preconfiguration register A
$D502	%W5B	MMU Preconfiguration register B
$D503	%W5T	MMU Preconfiguration register C
$D504	%W6B	MMU Preconfiguration register D
$D505	%W53	MMU ends configuration rgeister
$D50C	%W5A	MBU MMT configuration rgeister
$D507	%W5B	Page 0 pointer Lo
$D508	%W5B	Page 0 pointer Hi
$D509	%W5C	Page 1 pointer Lo
$D50A	%W5D	Page 1 pointer Hi
$D50B	%W5E	MMU version/reset register
$D600	%W7A	VDC Address register
$D601	%W8C	VDC data regTster
$E000-FC3B	%W2A-%W5T3	Kernel ROM
$FC3C FEFF	%W5A-%W6B	Unused ROM
$FF00-FF1C	%W6B-%W6A	FFU registers
$FF47 FFT3	%W6T-%W8B	Kernel jump tabTe
$FFFA FFFF	%W6A-%W6B	Hardware vectors

USEFUL BASIC INTERPRETER ADDRESSES

CPU SEX	C128 HEX	DESCRIPTION OF ROUTINE
$A00C	$A00C	Start vector
$A002		NMT vector
$A00Y		CBM BASIC
$A00C	$957C	Addresses of the BASIC commands minus T
$A00C	$9780	Addresses of the BASIC TuncTions
$A00C	$9A20	Hierarchy codes and addresses of the BASIC operators
$A00C	$9A17	Tist of BASIC command words
$A15E	$A04B	BASIC error messages
$A38A		Messages of the BASIC interpreter
$A38B	$A7B0	Stack Search-routine Ter FOR-NEXT and GOSUB
$A3BB	$7CBA	STack-shifting routine
$A3F9	$A7FE	Check on space in stack
$A40B	$5B17	Make space in memory
$A430		Output of BAT of memory
$A437	$B53C	Output of error messages
$A480	$1DA6	Break vector
$A47A	$A837	Ready vector
$A4BB	$1BC3	Input waiting-loop
$A4BC	$A9E2	Clear and inserting program Times
$A533	$A7AF	Tim BASIC program Times SHow
$A55B	$AF53	Gets a line into input BuTTer
$A571		Output of String tne Temp
$A57B	$A38A	Change of a Time into Interpreter-code
$A613	$5B6A	Look Tor start BBCrean of a BASIC Time
$A62E	$5T8B	BASIC command NEW
$A651	$51FB	BASIC command CLR
$A68C	$525A	Set program pointer to BASIC start
$A69C	$5B62	BASIC-command TIBY
$A717	$57AT	Change interpreter ends to command word
$A7A2	$53F3	BASIC command FOR
$A7AE	$5AF6	Interpreter loop carries out BASIC commands
$A7E4	$A4BF	Execute next BASIC statement
$A7EB	$A7F4	Carries out BASIC command
$A81B	$5ACA	BASIC command RESTORE
$A82C	$A1C7	Interrupts program at pr$A00C stop-key
$A8BY	$1BCB	BASIC-command STOP
$A831	$1BCB	BASIC-command END
$A857	$5AAA	BASIC-command CONT
$A871	$5A08	BASIC command RUN

```
A8A3 B55E BASIC command GOSUB
tt8A6 B588 BASIC command POKE
A8B1 B5E2 RtSIC command RETURN
A8B4 B6ddr BASIC command DATA
A8B8 B5E2A2 Looks for next statement
B1B8 B5B5 Looks for next line
B4B2B B5EC5 RtSIC command IF
B1B3A B625B BASIC command REM
A8B4B B53A3 BASIC command GW
B1B8C B5B4B Looks for address of A BASIC item
B4B5 B53C8 BASIC command LET
B4A8B B553A BASIC command ABIWIB
B1BA8 B55AB BASIC command CMD
A8A8 B55B8 BASIC command PRINT
BtB1 B6B2E Output string
A8B3B Output empty accumulator (0% carriage right)
Bt8B B674B Error handling for INPUT
BA27B BBB1B BASIC command BET
BA8AE BBB1B BASIC command INPUTB
Bt1BE BBAEB BASIC command INPUT
B4BtB BtBBC tring INPUT prompt and handle input
BtC8E B6BA8 BASIC-command READ
BACEC Output "?extra ignored" and "?BAD trac start
tABIB BB7F4 BASIC command NEXT
BA2BB B77B7 Evaluate numeric expression
B4A8B B77DA Checks on numeric
BA8BE B77BB Checks on string
BA8BE Output at !type mismatch
BtB3E B77EE Evaluate expression
BtE83 B7B87 Get arithmetic term
BAEAB B7BEB Floating point constant for A1
BAEBt tB7B8 BASIC command SGT
BtE71 B7B5B Gets term in parenthesis
B4EE7 B7B56 Checks on parenthesis closed
BtEEA B75Bt Checks on parenthesis open
BAEEB B75EC Checks on comma
B4EFt B75tE Checks on characters in accumulator
BtE8B B78BC Output at "Syntax error
BtE2A B7B7B Gets variable
B4Et7 tABE7 Get variable reference
3AFtB 3tC8B BASIC command GO
3AtE3 3tC8t BASIC command AND
3AtEB 3tCB6 Compare operations
4888t 65A7B BASIC command DIM
B8AB3 B7AAF Search for or create variable descriptor
BB113 Checks for letter
BB11B B7B46 Creats new 7 byte descriptor
3B1B5 B78BA Return address of variable
B31B4 Calculates pointer to first array-element
3B1BA 3B1BA Floating point constant 3B7BB0
BB1tC 3B1BC Changes ttC to INTEGER
BB1AC B8tA7 Input and convert floating to integer
BBBtE B4t8t FAC integer
3A1CA 37CA8 Search for or create array
3B2A5 37B85 Output at "Bad subscript
BB2A8 B72EB Output of "Illegal quantity
B53AC Calculates array size
3B37D 3B8B8 BASIC-function FRE
BB35A B8C78 Integer to FAC
tB3AE 3B4DB BASIC function POS
BBBAA BB4DB Checks on direct-mode
BB5A8 Output of "Illegal direct
BB3AE Output at undef'd function
BB3B3 B847A BASIC-command DEF
BB3E1 B85Z8 Checks on FN syntax
BB3tA B853B BASIC function FN
BB4B5 B85At BASIC function STR$
B8A7B String assignment: calculate pointer on string
B8Y8Y B8664 FRESTRING string
4BtEY 36208 tileneth string memory space
B85ZE B8CtE Garbage collection: remove unneeded strings
B85BB tt3B3 Is current string highest in memory?
3B83B B87BB String concatenate +
C8t7A B87tC Transfer string to memory
B86A3 B877E String assignment(as ERESTA
B8BBB B87EB Limits entry from temp string stack
B88BC B88tE BASIC Function CHR$
3B7BB 3B5BB RtSIC Function LEFT$
B87CC B8A8C BASIC function RIGHT$
BB737 B8B1C BASIC function MID$
B87BI B8A1B Pull string parameters GFE stack
B877C B8AB8 BASIC Function LEN
B87BB B8B1C Get string parameter
B8738 B8877 BASIC function ASC
B87BB 3B7t1 Sort Byte form 10-255)
B87tt B8Bt4 BASIC-function VAL
B87tB 88883 Gets address (0-65500) and Byte value (0-BA1
BB7E7 Change FAC to address-format (bangs 0-65535)
3BAB8 BA8C5 BASIC function PEEK
BBA8t BBA8B BASIC command POKE
BBA8B 8BC1B BASIC command WAIT
BBA1B B488E FAC = FAC + B 5
4885B B3B2E Minus FAC = constant (A/Y)  FAC
B8B5t 88B3t Minus tAC = BA8  FAC
B8B87 B8tt5 tiys FAC = constant 10/Y1 - FAC
B8B8C B8B1B tiys tAC = 488 = FAC
B8B47 B8B4C Complement FAC
BB87E 86550 Output at  Overflow
B8B83 B88AE Single byte multiply
B8B3C tB5BC Floating point constant for L88
B83tt BDA8t BASIC-function L88
tBB2B B8tB8 Multiplication FAC = constant (8/Y1 = FAC
B8tBB tB4B8 Multiplication FAC = 488 = FAC
B8B8C B8tB5 ABS = constant 1A/Y)
B8A87 B8A1C Add exponent tAC to Exponent of 488
B8tEE tB817 FAC = FAC = 1B
B8AEB B88FE Floating point constant 10
3Bttt BBBBB FAC = FAC/1B
BBB87 B83t Divide B88 by memory
AB1B7 B8tB8 tAC = constant 1t/Y1 / tAC
BBB12 B83tC tAC = B88/FAC
B8B4B BBB33 Output at "Division by zero

BBBA2 B8B05 FAC = constant
B8BC7 BA1EE Memory = tAC
B8BCA B8BFC Memory = FAC
BB8A8 B8C88 Variable = tAC
B8BFC B8C28 tAC = B88
B8C3C B8C38 tB5 = FAC
B8C8Y B8C36 Move FAC to ARG
B8C1B B8C47 Rounds FAC
B8C88 B8C57 Get sign of FAC
B8C78 B8C55 BASIC's Position B88
B8C58 B8C8% BASIC function B88
B8C5B BA1EE Compares constant 1B/Y) with FA
B8C5B B4CC7 Changes Erom FAC to integer
B8CCC BA81E BASIC function INT
B8C5I A88t8 Change ASCII re floating point
B8B78 BA8Fh Get new ASCII digit
B8B83 B8B17 Floating point constant for floating t  to BA111
B8BCE B8C88 Output of line number at error message
B8B7B B8B3C Output of positive integer number (0-65535
B8B88 B8t4B Change tAC to ASCII Format
4BF11 3AF7E Floating point constant B 5
B8F16 B8CB1 Binary numbers for change of EtC to BA111
B8F71 B8F87 BASIC function SGN
B8F7B EAC = constant 1B/Y) to the power of EAC
B8F7B B8FC1 tAC = B88 to the power of EAC
B8F8% B8tt5 BASIC negation functions
B8F8F B5B85 floating point constant for Ert
B8F8B B8AE3 BASIC function tAt
```

VIC Edit BOOKBASICS 88888-88887 1525%8-1525%1

```
  number8
HEX    DECIMAL  Bit   DESCRIPTION

B8888  132%8          Sprite B   X position (bits 0 B)
B8881  53241          Sprite B   Y position (bits B 1)
B888A  53238          Sprite 1   X position
B8883  53241          Sprite 1   Y position
B8884  53252          Sprite 2   X position
B8885  53253          Sprite 2   Y position
B8886  53238          Sprite 3   X position
B8887  53255          Sprite 3   Y position
B8888  53238          Sprite 4   X position
B8889  B8857          Sprite 4   Y position
B888A  B8258          Sprite 5   X position
B888B  53256          Sprite 5   Y position
B888C  53268          Sprite 6   X position
B888D  53261          Sprite 6   Y position
B888E  53888          Sprite 7   X position
B888B  B3263          Sprite 7   Y position
B8818  53269          9th bit of sprite X co-ordinate
                 0    Sprite B
                 1    Sprite 1
                 2    Sprite 2 are through sprite 7
                      VIC Control Register
B8812  53885          Raster compare register  Bit 8
                 7    Raster compare register  Position of raster on screen
B8813  53887          Light pen X position
B8814  53888          Light pen Y position
B8811  52288          Enable or B sprite sprite
                 B    1=Enable sprite B
                 1    1=Enable sprite 1
                 2    1=Enable sprite 2 sta through sprite 7
B8816  13878          VIC Control Register
                 4    1=Multicolour mode on
                 3    1=48 Column text B=B8 column text
                 2 B  Smooth scroll to X position
B8817  13871          Sprite Vertical Expansion
                 B    =Expd sprite B Vertically
                 1    =Expd sprite 1 Vertically
                 2    =Expd sprite 2 Vertically sta through to sprite 7
B881B  53272          VIC Memory Control
                 7-4  VIDEO matrix base address
                 3-B  Character set base address
B8819  53271          VIC Interrupt Eigns
                 7    Set to any VIC 188 condition
                 3    Light pen triggered (bit 7)
                 2    Sprite vs sprite triggered (bit 7)
                 1    Sprite vs background triggered (bit 7)
                 B    Raster compare triggered (bit 7)
B8814  53274          VIC Interrupt Switches
                 3    1=Enable light pen interrupt
                 2    1=Sprite vs sprite enabled
                 1    1=Sprit vs background enabled
                 B    1=Raster compare enabled
B881B  53275          Sprite Priority Registers
                 B 7  Each bit determines correspondingsprite priority 1=Sprite background priority
B881C  53278          Sprite multi colour select
                 B 7  Each bit sets corresponding sprite to multicolour
B881B  13877  B 7     Sprite Horizontal Expansion
B881t  53879          Sprite vs sprite collision detection If any sprites is touching another bitin  the bits corresponding to both sprites are turned on
B81t1  53878          Sprite/background collision detection If sprite has bit test or background character  the relevant bit is set
```

```
D020  53280     Border colour
D021  53281     Background colour
D022  53282     Multi colour 1
D023  53283     Multi colour 2
D024  53284     Multi colour 3
D025  53285     Sprite multi colour
D026  53286     Sprite multi colour
D027  53287     Sprite 0 colour
D028  53288     Sprite 1 colour
D029  53289     Sprite 2 colour
D02A  53290     Sprite 3 colour
D02B  53291     Sprite 4 colour
D02C  53292     Sprite 5 colour
D02D  53293     Sprite 6 colour
D02E  53294     Sprite 7 colour
```

USEFUL BASIC DATA STORAGE LOCATIONS

```
0B2C8-0BFE  704   706     Sprite block 11
0C340-0C7E  832   894     Sprite block 13
0C380-0CBC  896   940     Sprite block 14
0CBC0-0CFE  960-1000      Sprite block 15
```

SID CHIP ADDRESSES D400-D41C (54272-54300)

ADDRESS HEX	DECIMAL	BIT	DESCRIPTION
D400	54272		Voice 1 low byte of frequency
D401	54273		Voice 1 high byte of frequency
D402	54274		Voice 1 low byte of pulse width
D403	54275	3-0	Voice 1 High byte of pulse width
D404	54276		Voice 1 Control Register
		7	1=Random noise
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable voice 1
		2	1=Ring modulate voice 1 with voice 3
		1	1=Synchronize voice 1 with freq of voice 3
		0	1=Start attack decay cycle 0=Start release
D405	54277		Voice 1 Attack/Decay
		7-4	Attack cycle duration
		3-0	Decay cycle duration
D406	54278		Voice 1 Sustain/release
		7-4	Sustain cycle duration
		3 0	Release cycle duration
D407	54279		Voice 2 low byte of frequency
D408	54280		Voice 2 high byte of frequency
D409	54281		Voice 2 low byte of pulse width
D40A	54282	3-0	Voice 2 High byte of pulse width
D40B	54283		Voice 2 Control Register
		7	1=Random noise on
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable oscillator 2
		2	1=Ring modulate oscillator 2 with oscillator 1
		1	1=Synchronize oscillator 2 with oscillator 1 frequency
		0	1=Start attack decay cycle 0=Start release
D40C	54284		Voice 2 Attack/Decay
		7-4	Attack cycle duration
		3-0	Decay cycle duration
D40D	54285		Voice 2 Sustain/release
		7-4	Sustain cycle duration
		3-0	Release cycle duration
D40E	54286		Voice 3 low byte of frequency
D40F	54287		Voice 3 high byte of frequency
D410	54288		Voice 3 low byte of pulse width
D411	54289	3 0	Voice 3 high byte of pulse width
D412	54290		Voice 3 Control Register
		7	1=Random noise on
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable voice
		2	1=Ring modulate oscillator 3 with oscillator 2 output
		1	1=Synchronize oscillator 3 with freq of oscillator 2
		0	1=Start attack decay cycle 0=Start release
D413	54291		Voice 3 Attack/Decay
		7-4	Attack cycle duration
		3-0	Decay cycle duration
D414	54292		Voice 3 Sustain/release
		7 4	Sustain cycle duration
		3-0	Release cycle duration
D415	54293	2-0	Filter cut-off low nybble
D416	54294		Filter cut off high byte
D417	54295		Filter Control
		7 4	Filter resonance
		3	1=External input to filter
		2	1=Voice 3 to filter
		1	1=Voice 2 to filter
D418	54296	0	1=Voice 1 to filter
		7	Filter Volume and Mode
		6	1=Turn off voice 3 output
		5	1=high pass filter on
		4	1=band pass filter on
		3	1=Low pass filter on
		3-0	Output volume
D419	54297		A/D converter for paddle 1
D41A	54298		A/D converter for paddle 2
D41B	54299		Produces random number when voice 3 set to noise
D41C	54300		Output of voice 3 envelope generator

KERNAL ROM ROUTINES

JSR HEX	CLR HEX	Description of Routine
E043	A000	System 1 polynomial calculation
E059	E000	System 2 polynomial calculation
E3BF	8000	Floating point constants for BASIC
E3D7	8004	BASIC-function SGN
E407		Output of "Break"
E41C	9000	BASIC output of character
E41E	9005	BASIC receive a character
FF18	9A0A	CKOUT establish output-device
FF1E	9A0C	CKIN establish input-device
FF24	9A0D	GETIN get a character
E1D4	9A0E	BASIC-command SYS
FF56	9A12	BASIC-command SAVE
E1D5	9A29	BASIC command VERIFY
E156	9A2C	BASIC-command LOAD
FF1E	9A0A	BASIC-command OPEN
E1C7	9A15	BASIC-command CLOSE
E1DA	9A1C	99% parameters for LOAD/VERIFY/SAVE
E2F0	9A1A	99% integer in X
E386	9A1E	Get current char and check for line end
E3DE	9A1E	Check character for illegal values
E21B	9A1F	99% parameter for OPEN/CLOSE
E1E5	9A20	BASIC-function CHR$
E3B8	9A24	BASIC-function STR$
E1E9	9A2A	BASIC-function VAL
E2E6	9A2C	Floating point constants for CDS/SIN/TAN
E2E5	9A3A	2*PI in floating point
E2EA	9A3F	1/4 in floating point
E2EF	9A3A	More constants for CDS/SIN/TAN
E3BE	9A43	BASIC-function ATN
E3E1	9A42	Floating point constants for ATN
E378	9A80	BASIC NMI jump-in
E38B	9A1A	Error message handler
E394	9A82	BASIC cold start
E3A2	9A29	Copy of the CHRGET routine
E3A6		Start value for the RND function
E3BF	9A4F	Initialize RAM for BASIC
E3A7	9A57	Table of BASIC vectors
E3A3	9A5A	Load BASIC vectors
E3A5	9A1B	Messages of the operating system
E4E0	9C3F	Waits for Commodore key
E4EC		Constants for 60"B" timing
E500		Gets BASIC-address of CIA on VIA
E505		Gets screen format line/column
E50A	9CC8	Set cursor on get cursor position
E50A	9CF8	Screen reset
E544	9C1F	Clear screen
E56A	9C1B	Cursor home
E500		Initialize video controller
E5E0	9CBA	Set character from keyboard buffer
E5CA		Waiting loop for keyboard input
E632	9C2B	Get a character from the screen
E684	9C2F	Check for quote
E698		Calculate RAM for line start%
E59A		Table of colour codes
E5EA	9C3B	Scroll screen
E63C	9C46	Shift line up
E697	9C4E	Clear screen line
E6AC	9C75	Set character and colour on screen
E6A4		Calculate pointer on colour RAM
EA31	9F4E	Interrupt routine
EA87		Keyboard prompt
E8B4		Checks on SHIFT CTRL and CBM keys
E8F8	9D0F	Pointer on keyboard decoding tables
E8B1	9F4B	Decoding tables
E8C4		Check40 on control character
E8C9		Decoding tables
E8CB9	9E2C7	Constants for video controller
E8CE7		Load (or) Run (or)
E8CF8	9E77	LSB tables of screen starts
E8D9	E38B	Send TALK
F0BC	A6A3	Send LISTEN
E8F8	9E9E	Output of byte on IEC-bus
F0BD		Send secondary address for LISTEN
E8C7		Send secondary address for Talk
A8BF	9F1B	Send UNTALK
E8FE	9E5E6	Send UNLISTEN
E8E3	9EF5C	Get a byte from the IEC-bus
E8E7		One millisecond delay
A6C8	8E5F	Output 93E3F
E8FA	8E5E	Calculate number of 93E3F bits-bits
F0D4	9E75C	Output in 93E3F buffer
F0B6	9F7CE	GET of 93E3F
E4A4		Set timer for IEC time-out
F0B6		Error messages of the operating system
F128	9F7IE	Put out messages "
F157	9EFB6	Enable get a character
F16A	9F7B	OUTPUT output a character
F2EE	9E1E6	CKOUT tieing of the input-device
F250	9F1C	CLRCHT tieing of the output-device
F291	9F1B6	CLOSE
F30F		Look for logical file number

```
$F31F       Set file parameter
$F32F  AF22C CALF filename bit 1/8 channels
$F34A  AF2AF SPEN
$F34E  AF2B5 FIND
$F38F       Output  Searching for file name
$F3D2       Output  Loading/verifying
$F3EF  AF3E5 SAVE
$F3D7       Output  Saving filename
$F5ED  AF5FB UDTIM Increase real time clock
$F3E5  AF3EE Set files
$F3E4  AF3E6 Set file
$F3E8  AF3E8 Set file
$F3E8  AF3EF Test Stop key
$F3F5       Put out error message of the operating system
$F7EC  AF8C9 Read register number of file
$F7BA  AF81B Write header on tape
$F7F9       END. Start address of tape buffer
$F7B7       Set start and end address of the file buffer
$F7EB  AF89A Look for name on tape-header
$F80E  AF8AE Increase tape buffer pointer
$F817  AF8CA Waits for tape key for reading
$F82E       Find for tape key
$F81B  AF8CA Waits for tape key for writing
$F8A1  AF8F2 Read block of file
$F8A8  AF8FB Test Eighth EOF flag
$F8B4  AF91E Write tape buffer to tape
$F8E8  AF91B Write block on program on tape
$F8EE  AF A7E Wait for I/O end
$F8E1       Checks on stop key
$F9EC  AFA1B Interrupt routine for file read
$F9F7  AF95A Buf Bit counter for serial output
$F9A6  AF95B Write end Bit for tape
$F9C8  AF9DE Interrupt routine for file write
$FCEB  ACE27 Buf IRQ vector
$FCCF  AEE9B Switch off file drive
$FC81       Checks on reaching of end address
$FE1B       Increase address pointer
$FCE2  AFF3E RESET
$FB9E       Checks on ROM in A000 to C000
$FA1B       ROM module identification
$FA15       Set bi gap hardware and I/O vectors
$FB3B       Table of hardware and I/O vectors
$FD5B       Initialize work memory
$FD81  AFD1B Table of IRQ vectors
$FD1F       Buf parameter for file names
$FE0B       Set parameter for active file
$FF87       Put status
$FF1B       Set flag for messages of the operating system
$FE1C       Set status
$FE21       Buf Timeout flag for IEC bus
$FE25       Set or get RAM-upper limit
$FE34       Buf bi get RAM lower limit
$FE43  AFA1B NMI routine
$FE2C  AE85B Centronics for RS232 baud rate
$FFA8       Interrupt handler
```

C64 KERNAL JUMP TABLE

ADDRESS	CONTENTS	PURPOSE
$FF81	JMP $FF84	Initialize CIA's
$FF84	JMP $FF5B	Clear bi check RAM
$FF8A	JMP $FF15	Initialize I/O
$FF8D	JMP $FF1A	Initialize I/O vectors
$FF90	JMP $FE1B	Set status
$FF93	JMP $EE00	Send LISTEN on secondary address
$FF96	JMP $EE0C	Send TALK Secondary address
$FF99	JMP $FE25	SOL/bar RAM end
$FE1C	JMP $FE34	Set/get RAM status
$FF1F	JMP $E5A7	Scan keyboard
$FF84	JMP $FE01	Set IEC bus time out flag
$FFA5	JMP $EE13	Input for IEC-bus
$FF90	JMP $EE3B	Output to IEC bus
$FFA8	JMP $EE1E	Send UNTALK
$FFAB	JMP $EE1E	Send UNLISTEN
$FFB1	JMP $ED0C	Send LISTEN
$FFB4	JMP $EE2B	Send TALK
$FFB7	JMP $FE07	Set status
$FFB0	JMP $FE00	Set File parameter
$FFBA	JMP $FE0E	Set Filename parameter
$FFC0	JMP (0301)	$F264 OPEN
$FFC3	JMP (0303)	$F291 CLOSE
$FFC6	JMP (0305)	$F20E CHKIN set input device
$FFC9	JMP (0307)	$F250 CKOUT Set output device
$FFCC	JMP (0309)	$F333 CLRCH
$FFCF	JMP (030B)	$F157 BASIN Input character
$FFD2	JMP (030D)	$F1CA BSOUT output character
$FFD5	JMP $F49E	LOAD
$FFD8	JMP $F5ED	SAVE
$FFDB	JMP $FE0F	Set Time
$FFDE	JMP $FE0B	Set Time
$FFE1	JMP (0328)	$F6ED Scan stop-key
$FFE4	JMP $E09A)	$F13E GET
$FFE7	JMP (032C)	$F32F CLALF
$FFEA	JMP $F639	Increase Time
$FFED	JMP $E9ED	SCREEN get number lines and columns
$FFF0	JMP $E50A	Set/get screen position
$FFF3	JMP $E500	Get start of I/O element
$FFFA	JMP $FE43	NMI vector
$FFFC	JMP $FCE2	RESET vector

SCREEN COLOUR CODES And MODES

Values in PEEK For each colour

COLOUR	LOW NYBBLE VALUE	HIGH NYBBLE VALUE	MULTI COLOUR
Black	0	0	0
White	1	16	8
Red	2	32	10
Cyan	3	48	11
Purple	4	64	12
Green	5	80	13
Blue	6	96	14
Yellow	7	112	15
Orange	8	128	
Brown	9	144	
Light red	10	160	
Dark grey	11	176	
Mid grey	12	192	
Light green	13	208	
Light Blue	14	224	
Light grey	15	240	

Where to POKE colour values For each mode

MODE (1)	BIT 00 BIT PAIR	LOCATION	COLOUR VALUE
Background	0	53281	Low nybble
	1	Colour memory	Low nybble
Multicolour	00	53281	Low nybble
text	01	53282	Low nybble
	10	53283	Low nybble
	11	Colour memory	Multicolour
Extended	00	53281	Low nybble
colour text	01	53282	Low nybble
(1)2	10	53283	Low nybble
	11	53284	Low nybble
Bitmap	0	Screen memory	Low nybble (1)3
	1	Screen memory	High nybble (1)3
Multicolour	00	53281	Low nybble (1)4
Bitmap	01	Screen memory	High nybble (1)4
	10	Screen memory	Low nybble (1)4
	11	Colour memory	Low nybble

(1)2 For all modes the screen border colour is controlled by POKEing 53280 with the low nybble colour value

(1)3 In extended colour mode Bits 6 & 7 of each byte of screen memory serve as the bit-pair controlling background colour. Because only bits 0-5 are available for character selection, only characters with screen codes 0-63 can be used in this mode

(1)4 In the bitmapped modes the high and low nybble colour values are ORed together and POKEd into the SAME LOCATION in screen memory to control the colours of the corresponding CELL in the bitmap. For example to control the colours of cell 0 of the bitmap, OR the high and low nybble values and POKE the result into location 0 of screen memory

C128 COLOUR CODES

Command COLOR source colour

SOURCE NUMBER	SOURCE
0	40 column background colour
1	Foreground for Graphics screen
2	Foreground for multicolour 1
3	Foreground for multicolour 2
4	40-column border (text and graphics)
5	Text colour for 40- or 80-column screen
6	80-column background colour

40-COLUMN MODE COLOUR VALUE	COLOUR	80 COLUMN MODE COLOUR VALUE	COLOUR
1	Black	1	Black
2	White	2	White
3	Red	3	Red
4	Cyan	4	Light cyan
5	Purple	5	Light purple
6	Green	6	Light green
7	Blue	7	Dark Blue
8	Yellow	8	Light yellow
9	Orange	9	Dark purple
10	Brown	10	Brown
11	Light red	11*	Light red
12	Dark grey	12	Dark cyan
13	Medium grey	13	Medium grey
14	Light green	14	Light green
15	Light Blue	15	Light Blue
16	Light grey	16	Light grey

STANDARD CBM TOKENS

HEX	DEC	TOKEN	HEX	DEC	TOKEN	HEX	DEC	TOKEN
$00	0	END OF LINE	$40	64	@	$80	128	END
$01	1		$41	65	A	$81	129	FOR
$02	2		$42	66	B	$82	130	NEXT
$03	3		$43	67	C	$83	131	DATA
$04	4		$44	68	D	$84	132	INPUT#
$05	5		$45	69	E	$85	133	INPUT
$06	6		$46	70	F	$86	134	DIM
$07	7		$47	71	G	$87	135	READ
$08	8		$48	72	H	$88	136	LET
$09	9		$49	73	I	$89	137	GOTO
$0A	10		$4A	74	J	$8A	138	RUN
$0B	11		$4B	75	K	$8B	139	IF
$0C	12		$4C	76	L	$8C	140	RESTORE
$0D	13		$4D	77	M	$8D	141	GOSUB
$0E	14		$4E	78	N	$8E	142	RETURN
$0F	15		$4F	79	O	$8F	143	REM
$10	16		$50	80	P	$90	144	STOP
$11	17		$51	81	Q	$91	145	ON
$12	18		$52	82	R	$92	146	WAIT
$13	19		$53	83	S	$93	147	LOAD
$14	20		$54	84	T	$94	148	SAVE
$15	21		$55	85	U	$95	149	VERIFY
$16	22		$56	86	V	$96	150	DEF
$17	23		$57	87	W	$97	151	POKE
$18	24		$58	88	X	$98	152	PRINT#
$19	25		$59	89	Y	$99	153	PRINT
$1A	26		$5A	90	Z	$9A	154	CONT
$1B	27		$5B	91	[	$9B	155	LIST
$1C	28		$5C	92	£	$9C	156	CLR
$1D	29		$5D	93	]	$9D	157	CMD
$1E	30		$5E	94	↑	$9E	158	SYS
$1F	31		$5F	95	←	$9F	159	OPEN
$20	32	SPACE	$60	96		$A0	160	CLOSE
$21	33	!	$61	97	a	$A1	161	GET
$22	34	"	$62	98	b	$A2	162	NEW
$23	35	#	$63	99	c	$A3	163	TAB(
$24	36	$	$64	100	d	$A4	164	TO
$25	37	%	$65	101	e	$A5	165	FN
$26	38	&	$66	102	f	$A6	166	SPC(
$27	39	'	$67	103	g	$A7	167	THEN
$28	40	(	$68	104	h	$A8	168	NOT
$29	41	)	$69	105	i	$A9	169	STEP
$2A	42	*	$6A	106	j	$AA	170	+ (PLUS)
$2B	43	+	$6B	107	k	$AB	171	- (MINUS)
$2C	44	,	$6C	108	l	$AC	172	* (MULTIPLY)
$2D	45	-	$6D	109	m	$AD	173	/ (DIVIDE)
$2E	46	.	$6E	110	n	$AE	174	↑ (POWER)
$2F	47	/	$6F	111	o	$AF	175	AND
$30	48	0	$70	112	p	$B0	176	OR
$31	49	1	$71	113	q	$B1	177	> (GREATER)
$32	50	2	$72	114	r	$B2	178	= (EQUAL)
$33	51	3	$73	115	s	$B3	179	< (LESS)
$34	52	4	$74	116	t	$B4	180	SGN
$35	53	5	$75	117	u	$B5	181	INT
$36	54	6	$76	118	v	$B6	182	ABS
$37	55	7	$77	119	w	$B7	183	USR
$38	56	8	$78	120	x	$B8	184	FRE
$39	57	9	$79	121	y	$B9	185	POS
$3A	58	:	$7A	122	z	$BA	186	SQR
$3B	59	;	$7B	123	{	$BB	187	RND
$3C	60	<	$7C	124	|	$BC	188	LOG
$3D	61	=	$7D	125	}	$BD	189	EXP
$3E	62	>	$7E	126	~	$BE	190	COS
$3F	63	?	$7F	127	○	$BF	191	SIN
$40	64	@	$80	128		$C0	192	TAN
$41	65	A	$81	129		$C1	193	ATN
$42	66	B	$82	130		$C2	194	PEEK
$43	67	C	$83	131		$C3	195	LEN
$44	68	D	$84	132		$C4	196	STR$
$45	69	E	$85	133		$C5	197	VAL
$46	70	F	$86	134		$C6	198	ASC
$47	71	G	$87	135		$C7	199	CHR$
$48	72	H	$88	136		$C8	200	LEFT$
$49	73	I	$89	137		$C9	201	RIGHT$
$4A	74	J	$8A	138		$CA	202	MID$
$4B	75	K	$8B	139		$CB	203	GO
$4C	76	L	$8C	140				
$4D	77	M	$8D	141				
$4E	78	N	$8E	142				
$4F	79	O	$8F	143				
$50	80	P	$90	144				
$51	81	Q	$91	145				
$52	82	R	$92	146				
$53	83	S	$93	147				
$54	84	T	$94	148				
$55	85	U	$95	149				
$56	86	V	$96	150				
$57	87	W	$97	151				
$58	88	X	$98	152				
$59	89	Y	$99	153				
$5A	90	Z	$9A	154				
$5B	91	[	$9B	155				
$5C	92	£	$9C	156				
$5D	93	]	$9D	157				
$5E	94	↑	$9E	158				
$5F	95	←	$9F	159				

CBM64 EXTENDED TOKENS

HEX	DEC	TOKEN	HEX	DEC	TOKEN	HEX	DEC	TOKEN
$CC	204	DEC	$E0	224	FILTER	$FE	254	DIRECTORY
$CD	205	HEX$	$E1	225	GRAPHIC	$FF	255	BANK
$CE	206	reserved	$E2	226	PAINT	$100	256	BLOAD
$CF	207	JOY	$E3	227	CHAR	$101	257	BSAVE
$D0	208	DMOT	$E4	228	BOX	$102	258	SCRATCH
$D1	209	DEC	$E5	229	CIRCLE	$103	259	COLLECT
$D2	210	HEX$	$E6	230	GSHAPE	$104	260	COPY
$D3	211	ERR$	$E7	231	SSHAPE	$105	261	SCHAVE
$D4	212	INSTR	$E8	232	DRAW	$106	262	BACKUP
$D5	213	RLIE	$E9	233	LOCATE	$107	263	DELETE
$D6	214	RESUME	$EA	234	COLOR	$108	264	RENUMBER
$D7	215	TRAP	$EB	235	SCNCLR	$109	265	KEY
$D8	216	TRON	$EC	236	SCALE	$10A	266	MONITOR
$D9	217	TROFF	$ED	237	HELP	$10B	267	USING
$DA	218	GOTO	$EE	238	DO	$10C	268	UNTIL
$DB	219	USL	$EF	239	LOOP	$10D	269	WHILE
$DC	220	MUTE	$F0	240	EXIT	$10E	270	1000reset

CBM128 DOUBLE BYTE TOKENS

BCE followed by

HEX	DEC	TOKEN	HEX	DEC	TOKEN	HEX	DEC	TOKEN
$06	2	POT	$06	5	ASPTOB	$00	0	XOR
$03	3	BUMP	$06	6	ASPRITE	$00	0	$0000000
$04	4	PEN	$07	7	RSPCOLOR	$06	10	POINTER

BFE followed by

HEX	DEC	TOKEN	HEX	DEC	TOKEN	HEX	DEC	TOKEN
$03	2	BANK	$0E	14	APPEND	$10	27	$007
$03	3	FILTER	$FF	15	DCLOSE	$1C	28	DISVE
$04	4	PLAY	$10	16	BSAVE	$10	29	SPRSAV
$05	5	TEMPO	$11	17	BLOAD	$1E	30	BUIT
$06	6	MOVSPR	$12	18	SECCRA	$1F	31	SPRSAV
$07	7	SPRITE	$13	19	CONCAT	$20	00	Built
$08	8	BOXCOLOR	$14	20	DVERIFY	$21	33	STASH
$09	9	XRAY	$15	21	DCLEAR	$2F	34	FETCH
$0A	10	ENVELOPE	$10	00	SPRSAV	$23	35	$0AC
$0B	11	SLEEP	$17	23	COLLISION	$24	36	OFF
$0C	12	CATALOG	$18	24	$0BIN	$06	37	FWDT
$0D	13	DOPEN	$19	25	MOND	$26	00	SLOW
			$10	26	DINMON			

1541 810K DRIVE - USEFUL MEMORY LOCATIONS

$00 $0000$$

HEX	DECIMAL	DESCRIPTION
$0000-007F	0-2047	906 RAM CHIP
$0000	0	Command $0$$ for buffer 0
$0001	1	Command code for buffer 1
$0004	4	Command code for buffer f
$0005	5	Command code for buffer 3
$0006	6	Command $0000 for buffer 4
$0006-0007	6-7	Track and sector for buffer 0
$0008-0009	8-9	Track and sector for buffer 1
$000A-000B	10-11	Track and sector for buffer 2
$000C-000D	12-13	Track and sector for buffer 3
$000E-000F	14 15	Track and sector for buffer 4
$0010-0013	16 15	ID for drive 0
$0014-0015	20 21	ID for drive 1
$0016 0017	22-23	Current ID
$0018-0021	24 33	Flag for head irrespect
$0022-0031	34-48	Buffer pointer for disk controller
$0036	07	Contains 0 - mark for $0$$ing of data block header
$0025A	00	Parity for data buffer
$0038	01	Drive number for disk controller
$003F	63	Buffer number for disk controller
$0043	07	Number of sectors per track for formatting
$0007	71	Contains 7 - mark for begining of data block header
$0049	73	Stack pointer
$004A	74	Step number for head irrespect
$0001	97	Actual track number for formatting
$0053	105	Step size for motor $ivision (=10)
$005A	100	Number of read attempts 15)
$005F-0070	111 112	Pointer to address for 0 and 8 commands
$0077	119	Device number+ 920(32) for LISTIN
$0078	120	Device number+ $+8($1 for TALK
$0079	121	Flag for LISTEN 1)/0)
$007A	122	Flag for TALK 1)/0)
$007C	124	Flag for ATN from serial bus receiving
$007B	125	Flag for EOI from serial bus
$007F	127	Drive Number (0)
$0080	128	Current track number
$0081	129	Current sector number
$0082	130	Current channel number
$0083	131	Current File Number
$0084	132	Current secondary address
$0085	133	Current $ata byte
$0088-008B	136-141	Work storage for $ivision
$009F-009$	140-159	Actual buffer pointer
$0000-0008	153 184	Address of buffer 0 (0$00$)
$0000-009C	155 156	Address of buffer 1 (09$0$)
$0000-009E	157 158	Address of buffer 2 (0$$$0$)
$0000F-0000	159 160	Address of buffer 3 (0$$0$1
$0000-00A2	161-100	Address of buffer 4 (0$$$2$)
$00A3 00A4	163-164	Pointer to input buffer $0200
$00A5 0000	165 166	Pointer to buffer error message ($0$$5)
$0000 0000	181 100	Record number LO block number LO
$0000 00CH	107-100	Record number HI block number HI
$00C1-00C$	153 100	Write pointer low REL file
$00C7-00CC	100 204	Record length low REL file
$0000	515	Pointer in lo$e$ for SEL file
$0000	215	Side sector number
$0000	214	Pointer to data block in side sector
$0007	51%	Pointer to record in REL file
$00E7	231	File type
$0000	240	Buffer number
$0100-0145	256-006	Stack
$0200-0059	515 552	Buffer for command string
$025A	506	File type
$0258	006	Record length
$0259	001	Track side sector
$025A	002	Sector side sector
$0274	000	Length of input line
$0278	0 $0	Number of file names
$0297	663	File control values
$0200-0000	006-041	Track BF a file
$0205-0208	045-049	Sector BF a file
$0000-0218	725 781	Buffer for $ider messages
$0000-02FC	782 784	Number of BLOCKS FREE
$0300 03FF	700-1003	Buffer 0 main work buffer
$0000-04FF	1024 1279	Buffer 1 disk $irectory
$0500-05FF	1280-1535	Buffer 5 - user buffer
$0000-0000	1536 1791	Buffer 3 disk $irectory
$0700-07F1	1792 2047	Buffer 4 RAM 000
$0000 11FF	2048-6143	Unused
$1000 1000	5144-6100	IEEE 000 controller 6522
$1010 10FF	6100 7107	Unused
$1E00 1C0F	7100-7103	Drive controller 6522
$1C10-C0FF	7104-49407	Unused
$C100-FFFF	49000 65535	High operating system routines

1541 DISK ERROR MESSAGES AND THEIR CAUSES

The following list contains the error messages reassigned by the 1541 DOS
Note that TT and SS denote Track and Sector respectively

ERROR NUMBER	DESCRIPTION
00 OK 00 00	The last disk operation was error free or no disk command has been made since the last error message was read
20 READ ERROR TT SS	The header of a block was not found. It is usually the result of a defective disk. TT and SS denote the track and sector in which the error occurred Remedy: Reinit the disk
21 READ ERROR TT 00	The SYNC marker of a block was not found. The cause may be an unformatted disk or no disk in the drive. This error can also be caused by a misaligned read/write head Remedy: Reinit, insert a disk and realign it if necessary or have the head re-aligned
22 READ ERROR TT SS	A checksum error has occurred in the header of a data block, which may have been caused by the incorrect writing of a block or rough handling of the disk
23 READ ERROR TT SS	A data block was read into the DOS buffer but a checksum error has occurred. One or more data bytes are incorrect Remedy: Save as many files as possible onto another disk
24 READ ERROR TT SS	This error also results from a checksum error in the data block or in the preceding data header. Incorrect bytes have been read Remedy: Same as for error 23
25 WRITE ERROR TT SS	This is actually a VERIFY error. After writing every block the data is read again, checked against the data in the buffer. This error is produced if the data are not identical Remedy: Format the cassette that caused the error. If this does not work, the block allocate command can be used to look out the defective block for future use
26 WRITE PROTECT ON TT SS	An attempt was made to write to a disk with a write protect tab on Remedy: Remove the tab
27 READ ERROR TT SS	A checksum error has occurred in the header of a data block Remedy: Repeat command or remove disk
28 WRITE ERROR TT SS	After writing a data block the SYNC characters of the next data block were not found Remedy: Format the disk again or exchange it
29 DISK ID MISMATCH TT SS	The ID in the DOS memory does not agree with the ID on the disk. The disk either was not initialised or has an error in the header of a data block Remedy: Initialise the disk
30 SYNTAX ERROR 00 00	The DOS cannot understand the command that it is receiving Remedy: Correct the command
31 SYNTAX ERROR 00 00	A command was not recognised by the DOS Remedy: Do not use the command
32 SYNTAX ERROR 00 SS	The command sent was over 58 characters long Remedy: Shorten the command
33 SYNTAX ERROR 00 00	A wildcard (* or ?) was used in an OPEN or SAVE command Remedy: Remove wildcard
34 SYNTAX ERROR 00 00	The DOS cannot find the filename in a command. The cause may be a forgotten colon after the command word Remedy: Check the command
39 FILE NOT FOUND 00 00	When program (USR) was not found. Error automatic execution Remedy: Check filename
50 RECORD NOT PRESENT 00 00	A non-existent record was addressed in a relative data file, when writing a record this is not really an error Remedy: You can avoid this message if you write CHR$(255) with the highest record number when initialising the file
51 OVERFLOW IN RECORD 00 00	The number of characters sent when writing a record in a relative file was greater than the record length. The excess characters are ignored
52 FILE TOO LARGE 00 00	The record number within a relative file is too big, the disk does not have enough capacity Remedy: Use another disk or reduce the number of records
60 WRITE FILE OPEN 00 00	An attempt was made to OPEN a file and not previously been CLOSE: after writing Remedy: Use dana h in the OPEN command in read the file
61 FILE NOT OPEN 00 00	Records was attempted in a file that has not been OPENed Remedy: OPEN the file as check the filename
62 FILE NOT FOUND 00 00	An attempt was made to load a program or open a file that does not exist on the disk Remedy: Check the filename
63 FILE EXISTS 00 00	An attempt was made to establish a new file with the same name as one already on the disk Remedy: Use a different name as use 00
64 FILE TYPE MISMATCH 00 00	The file type used in the OPEN command does not agree with the file type in the directory Remedy: Correct the filetype
65 NO BLOCK TT SS	This message is given in association with the block-allocate command when the specified block is no longer free. In this case the DOS automatically searches for a free block with a higher sector and/or track number and gives these values as the track and sector number in the error message. If no block with a greater number is free two zeros will be given
66 ILLEGAL TT or 00 TT 00	An attempt has been made to access a non-existent block using the block commands
67 ILLEGAL TT or 00 TT SS	The track/sector combination of a file contains values for a non-existent track or sector
70 NO CHANNEL 00 00	An attempt has been made to open more files whereas then are available or a channel access channel is already reserved Remedy: Block closes a channel after it has been accessed
71 DIR ERROR TT 00	The number of free blocks in the DOS memory does not agree with the BAM. Often this means the disk has not been initialised Remedy: If the disk has been initialised validate it
72 DISK FULL 00 00	Fewer than three blocks are free on the disk or the maximum number of directory entries have been used (144 on the 1541) Remedy: Use a different disk or try validating to free any blocks that may be available
73 CBM DOS v 00 1541 00 00	The message is the power-up message of the 1541. It appears as an error message when an attempt is made to write to a disk that has not formatted with the same DOS version
74 DRIVE NOT READY 00 00	The drive does not have a disk inserted
75 FORMAT SPEED ERROR 00 00	This error only occurs on the CBM 8050

LOCATIONS 197 CBM KEYCODE VALUES

KEY	KEYCODE	KEY	KEYCODE
A	18	@	18
B	28	&	18
C	20	7	24
D	18	&	27
E	19	&	32
F	21	&	35
G	48	*	36
H	48		43
I	33	£	35
J	34	CLR/HOME	51
K	37	INST/DEL	0
L	42	LEFT ARROW	57
M	36	&	35
N	35	*	35
O	38		24
P	41		46
Q	35	↓	50
R	17	*	53
S	13	RET	1
T	35		47
U	30		44
V	31	/	50
W	9	CBM UP/DOWN	7
X	20	CBM LT/RT	0
Y	28	F1	4
Z	12	F3	5
1	56	F5	6
2	59	F7	3
3	8	SPACE	36
4	11	RUN/STOP	63

NO KEY PRESSED = 64

CBM VALUES FOUND AT LOCATION 653

CODE	DESCRIPTION
0	No key pressed
1	SHIFT
2	CBM
3	SHIFT and CBM
4	CTRL
5	SHIFT and CTRL
6	CBM and CTRL
7	SHIFT CTRL and CBM

8510 ADDRESSING MODES AND OPERATING CODES

The following table gives the hex values for the various opcodes in their individual addressing modes. The following key to be used for the Address Mode

A = Accumulator
# = Immediate
ZP = Zero page
ABS = Absolute
ABX = Absolute X
ABY = Absolute Y
ZPX = Zero page X
ZPY = Zero page Y
IX = Indexed X
I Y = Indexed Y

MNEMONIC	A	#	ZP	#8	ABX	ABY	ZPX	ZPY	IX	I Y
ADC	--	69	65	60	70	79	75	--	61	71
AND	--	29	25	20	30	39	35	--	21	31
ASL	36	-	06	0E	1E	--	16	--	--	-
BIT	-	--	24	2C	--	--	--	--	--	--
CMP	--	C9	C5	CD	DD	D9	D5	--	C1	D1
CPX	-	E0	E4	EC	-	--	--	--	--	-
CPY	-	C0	C4	CC	--	--	-	--	-	--
DEC	--	--	C6	CE	DE	--	D6	--	--	--
EOR	--	49	45	40	50	59	55	--	41	51
INC	--	--	E6	EE	FE	--	F6	--	-	--
LDA	--	A9	A5	AD	BD	B9	B5	--	A1	B1
LDX	--	A2	A6	AE	--	BE	--	B6	-	-
LDY	--	A0	A4	AC	BC	--	B4	-	--	--
LSE	4A	--	46	4E	5E	--	56	--	--	--
ORA	-	09	05	00	10	19	15	-	01	11
ROL	2A	--	26	2E	3E	--	36	--	--	--
ROR	6A	--	66	6E	7E	--	76	--	--	--
SBC	--	E9	E5	ED	FD	F9	F5	--	E1	F1
STA	--	--	85	8D	9D	99	95	--	81	91
STX	--	-	86	8E	--	--	--	96	--	-
STY	--	--	84	8C	-	--	94	--	--	--

GROUPED INSTRUCTIONS

Branch Instructions

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEQ
10	30	50	70	90	B0	D0	F0

Transfer Instructions

TXA	TAX	TYA	TAY	TSX	TXS
8A	AA	98	A8	BA	9A

Stack Instructions

PHP	PLP	PHA	PLA
08	28	48	68

Jump Instructions

JSR	RTI	RTS	JMP	JMP	#BP
20	40	60	4C	6C	EA

Flag Instructions

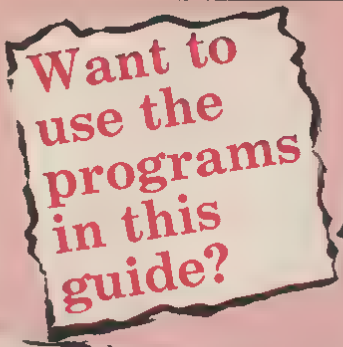
CLC	SEC	CLI	SEI	CLV	CLD	SED
18	38	58	FB	B8	D8	F8

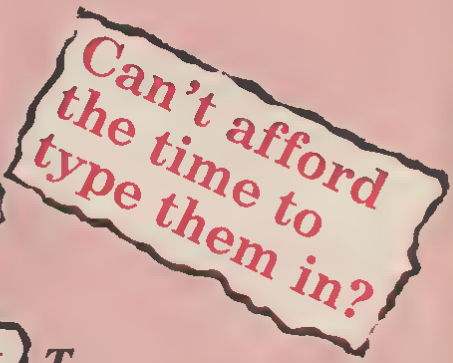
INC/DEC Instructions

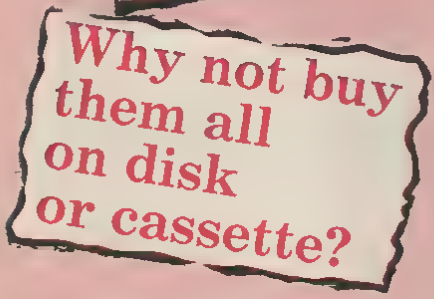
DEY	INY	DEX	INX
88	C8	CA	E8

HEX TO DECIMAL CONVERTER

HEX	DECIMAL LOW	DECIMAL HIGH	HEX	DECIMAL LOW	DECIMAL HIGH	HEX	DECIMAL LOW	DECIMAL HIGH
00	0	0	40	64	256	80	128	512
01	1	256	41	65	257	81	129	513
02	2	512	42	66	258	82	130	514
03	3	768	43	67	259	83	131	515
04	4	1024	44	68	260	84	132	516
05	5	1280	45	69	261	85	133	517
06	6	1536	46	70	262	86	134	518
07	7	1792	47	71	263	87	135	519
08	8	2048	48	72	264	88	136	520
09	9	2304	49	73	265	89	137	521
0A	10	2560	4A	74	266	8A	138	522
0B	11	2816	4B	75	267	8B	139	523
0C	12	3072	4C	76	268	8C	140	524
0D	13	3328	4D	77	269	8D	141	525
0E	14	3584	4E	78	270	8E	142	526
0F	15	3840	4F	79	271	8F	143	527
10	16	4096	50	80	272	90	144	528
11	17	4352	51	81	273	91	145	529
12	18	4608	52	82	274	92	146	530
13	19	4864	53	83	275	93	147	531
14	20	5120	54	84	276	94	148	532
15	21	5376	55	85	277	95	149	533
16	22	5632	56	86	278	96	150	534
17	23	5888	57	87	279	97	151	535
18	24	6144	58	88	280	98	152	536
19	25	6400	59	89	281	99	153	537
1A	26	6656	5A	90	282	9A	154	538
1B	27	6912	5B	91	283	9B	155	539
1C	28	7168	5C	92	284	9C	156	540
1D	29	7424	5D	93	285	9D	157	541
1E	30	7680	5E	94	286	9E	158	542
1F	31	7936	5F	95	287	9F	159	543
20	32	8192	60	96	288	A0	160	544
21	33	8448	61	97	289	A1	161	545
22	34	8704	62	98	290	A2	162	546
23	35	8960	63	99	291	A3	163	547
24	36	9216	64	100	292	A4	164	548
25	37	9472	65	101	293	A5	165	549
26	38	9728	66	102	294	A6	166	550
27	39	9984	67	103	295	A7	167	551
28	40	10240	68	104	296	A8	168	552
29	41	10496	69	105	297	A9	169	553
2A	42	10752	6A	106	298	AA	170	554
2B	43	11008	6B	107	299	AB	171	555
2C	44	11264	6C	108	300	AC	172	556
2D	45	11520	6D	109	301	AD	173	557
2E	46	11776	6E	110	302	AE	174	558
2F	47	12032	6F	111	303	AF	175	559
30	48	12288	70	112	304	B0	176	560
31	49	12544	71	113	305	B1	177	561
32	50	12800	72	114	306	B2	178	562
33	51	13056	73	115	307	B3	179	563
34	52	13312	74	116	308	B4	180	564
35	53	13568	75	117	309	B5	181	565
36	54	13824	76	118	310	B6	182	566
37	55	14080	77	119	311	B7	183	567
38	56	14336	78	120	312	B8	184	568
39	57	14592	79	121	313	B9	185	569
3A	58	14848	7A	122	314	BA	186	570
3B	59	15104	7B	123	315	BB	187	571
3C	60	15360	7C	124	316	BC	188	572
3D	61	15616	7D	125	317	BD	189	573
3E	62	15872	7E	126	318	BE	190	574
3F	63	16128	7F	127	319	BF	191	575
40	64	16384	80	128	320	C0	192	576
41	65	16640	81	129	321	C1	193	577
42	66	16896	82	130	322	C2	194	578
43	67	17152	83	131	323	C3	195	579
44	68	17408	84	132	324	C4	196	580
45	69	17664	85	133	325	C5	197	581
46	70	17920	86	134	326	C6	198	582
47	71	18176	87	135	327	C7	199	583
48	72	18432	88	136	328	C8	200	584
49	73	18688	89	137	329	C9	201	585
4A	74	18944	8A	138	330	CA	202	586
4B	75	19200	8B	139	331	CB	203	587
4C	76	19456	8C	140	332	CC	204	588
4D	77	19712	8D	141	333	CD	205	589
4E	78	19968	8E	142	334	CE	206	590
4F	79	20224	8F	143	335	CF	207	591
50	80	20480	90	144	336	D0	208	592
51	81	20736	91	145	337	D1	209	593
52	82	20992	92	146	338	D2	210	594
53	83	21248	93	147	339	D3	211	595
54	84	21504	94	148	340	D4	212	596
55	85	21760	95	149	341	D5	213	597
56	86	22016	96	150	342	D6	214	598
57	87	22272	97	151	343	D7	215	599
58	88	22528	98	152	344	D8	216	600
59	89	22784	99	153	345	D9	217	601
5A	90	23040	9A	154	346	DA	218	602
5B	91	23296	9B	155	347	DB	219	603
5C	92	23552	9C	156	348	DC	220	604
5D	93	23808	9D	157	349	DD	221	605
5E	94	24064	9E	158	350	DE	222	606
5F	95	24320	9F	159	351	DF	223	607
60	96	24576	A0	160	352	E0	224	608
61	97	24832	A1	161	353	E1	225	609
62	98	25088	A2	162	354	E2	226	610
63	99	25344	A3	163	355	E3	227	611
64	100	25600	A4	164	356	E4	228	612
65	101	25856	A5	165	357	E5	229	613
66	102	26112	A6	166	358	E6	230	614
67	103	26368	A7	167	359	E7	231	615
68	104	26624	A8	168	360	E8	232	616
69	105	26880	A9	169	361	E9	233	617
6A	106	27136	AA	170	362	EA	234	618
6B	107	27392	AB	171	363	EB	235	619
6C	108	27648	AC	172	364	EC	236	620
6D	109	27904	AD	173	365	ED	237	621
6E	110	28160	AE	174	366	EE	238	622
6F	111	28416	AF	175	367	EF	239	623
70	112	28672	B0	176	368	F0	240	624
71	113	28928	B1	177	369	F1	241	625
72	114	29184	B2	178	370	F2	242	626
73	115	29440	B3	179	371	F3	243	627
74	116	29696	B4	180	372	F4	244	628
75	117	29952	B5	181	373	F5	245	629
76	118	30208	B6	182	374	F6	246	630
77	119	30464	B7	183	375	F7	247	631
78	120	30720	B8	184	376	F8	248	632
79	121	30976	B9	185	377	F9	249	633
7A	122	31232	BA	186	378	FA	250	634
7B	123	31488	BB	187	379	FB	251	635
7C	124	31744	BC	188	380	FC	252	636
7D	125	32000	BD	189	381	FD	253	637
7E	126	32256	BE	190	382	FE	254	638
7F	127	32512	BF	191	383	FF	255	639
80	128	32768	C0	192	384			







Typing in long programs can be a pretty daunting task Once you've entered the program there will probably be typing errors that need to be corrected. Why not save yourself time and trouble by buying a disk or cassette containing all of the programs from this magazine at a bargain price of £6 00 (disk) or £4 00 (cassette)

The disk and cassette are only available by mail order from the address on the order form. A cheque payable to ASP Ltd for the correct amount should be included with the order Overseas customers should add £1 00 for postage

ORDER FORM — PLEASE COMPLETE IN BLOCK CAPITALS

NAME	QUANTITY	PRICE	ORDER CODE	TOTAL
SERIOUS USER DISK		£6 00	YSUGD2	
SERIOUS USER TAPE		£4 00	YSUGC2	
OVERSEAS POSTAGE		£1 00		
			TOTAL	

NAME .....

ADDRESS .....

..... POSTCODE .....

I enclose a cheque/postal order for £ .. made payable to ASP LTD for the Your Commodore Serious User Guide Disk/Tape

All orders should be sent to Your Commodore Readers Services, Argus Specialist Publications, 9 Hall Road, Hemel Hempstead, Herts HP2 7BH

Please allow 28 days for delivery

128 Font Editor

A character designer to turn your C128 into the definitive machine

by Julian Mackintosh

User-defined characters can turn a so-so screen into a work of art but their creation involves the programmer in hours of preparatory work. Font Editor makes the C128's brain take the strain by providing an environment which is easy to master and can be expanded as the user's needs dictate

Font Editor is a machine code program located between $3000 and $4000 (12288–16384), with two characters sets stored at $2000 (8192) and $2800 (10240) These locations are normally reserved for the screen in graphic modes 1, 2, 3 and 4, which means that the Font Editor can't be used alongside these four modes

Before the editor is used, it is important to understand how the C128 character sets are normally used The two sets are categorised as upper case with graphics, and upper case with lower case Only one set can be used at a time and they can be swapped by holding down the SHIFT key and pressing the CBM key

The character shapes are copied from a table in ROM but it is possible, by pointing the operating system towards an area of RAM to load in two user-defined character sets The 128 Font Editor has been written to help with the design of these alternative sets

Controlling the editor

The Font Editor screen is divided into several windows, or areas, which control all of the functions necessary for redefining the characters By using a joystick in Port 2, the screen pointer can be moved from area to area and, when the desired mode is highlighted, the fire button is pressed to select the new mode

The function of each window is shown in Diagram 1

The options

The 20 option menu allows complex operations, like mirroring and rotation of characters, to be performed at the touch of a button Tables 1 and 2 show the variety of the functions available through these menus The load and save operations only operate on the current character set, so the set has to be selected from the menu before these options are executed

These tables only show the standard features but complex manipulations can be achieved by using two or more options in sequence Any character can be flipped in the diagonal plane by first rotating the character through 90 or 270 degrees and then by flipping the character in the horizontal plane

The best way to execute these special operations is through user-defined routines written in Basic or code To execute such a function, select the End Editor option and the computer will return to the mode it was in when the editor was entered. For example, if the editor was called from within a program by a line such as

```
10 BOOT"FONT EDITOR"
```

it will return to program control at the next program line If it was entered from direct mode then the READY prompt will appear

When the defined routine has been executed, the Font Editor can be re-entered with SYS DEC ("382F") as long as the option screen has not been corrupted by the new routine

By using calls to the editor's code, it is possible to add extremely complex functions to the editor whenever your needs dictate

Defining new options

There are several things to bear in mind when defining an option The editor code and character definition areas must not be corrupted by the new routines, but the input window (Area 4) can be used To facilitate extended options, the function keys can be used when the program is in input mode but Area 4 should always be cleared before re-entering the editor

The RUN/STOP and RESTORE function is not disabled by the program but, if the program is interrupted by using these keys, it can be restarted by SYS 12288 If, however, the screen has been corrupted, the editor will have to be rebooted before this system call

Always, remember that the cur-

rent character set remains operative when the End Editor option is used. This also applies to whether the set is taken from ROM or RAM

The following breakdown of the Font Editor memory map will help when defining your own routines

$0A2C The lower nybble indicates the current source of the character set
%xxxx0100 ($4) for ROM set 1
%xxxx0110 ($6) for ROM set 2
%xxxx1000 ($8) for RAM set 1
%xxxx1010 ($A) for RAM set 2

$0C00 Start of the relocated interrupt routine for mating the cursor around the screen It appears here to avoid problems if the graphics screen is cleared.

$0D63 The X position of the pointer when the fire button was last pressed

$0D64 The Y position corresponding to $0D63

$0D65 Register for the currently selected menu option in Area 3 No option is selected if the value is $FF

$0D66 Flag to indicate that the fire button has been pressed This is reset from 1 to 0 when the signal is acknowledged

$3000 SYS called for a cold start only after RUN/STOP and RESTORE have been pressed at the same time

$30AE This routine clears the input window, Area 4

$30D6 Forces Area 1 to display the current character held in the accumulator It also updates Areas 2, 5 6 and 7 accordingly From Basic this is called by

SYS DEC("30D6"),xp

where xp is the screen poke value for the character to be displayed

$3260 This inverts the pixel at the location specified by the accumulator (column) and the Y register (row) From Basic.

SYS DEC("$3260"),c,r

where both variables are within the range zero to seven

$3662 Over-writes the RAM characters with the ROM set

$382F WARM START This is the call for restarting the editor when re-entering from a user-defined option Before calling this address remember to reset any windows that may have been used, especially Area 4

The following commands only affect the character indicated by Area 5 Any other characters which need to be altered must first be called to the screen by the routine at $30D6

$32D5 Clears the character in Area 1

$32EC Inverts the current character

$3306 Mirrors the bottom half of the current character into the top area.

$332D Mirrors the top of the current character into the bottom half

$3354 Flips the character in the horizontal plane

$3384 Rotates the character by 90 degrees.

$33D7 Mirrors the right half of the character onto the left side.

$340E Mirrors the left side of the character onto the right half

$3445 Flips the character in the vertical plane

$36A0 Copies one character from ROM into the current RAM slot

When the End Editor option is used, the pointer remains operative and can be used in the user-defined function

Redefining characters can be enjoyable and satisfying, the 128 Font Editor can increase the enjoyment if it is used wisely

Table 1 The Function Menu

COMMAND	*FUNCTION*
LOAD	*Loads a font from disk after allowing its name to be input via Area 4 The font is placed into the RAM locations for the currently selected character set*
SAVE	*The current character set is saved to disk after its filename is input through Area 4*
NEW ALL	*This replaces all of the current RAM characters with a copy of the corresponding ROM set*

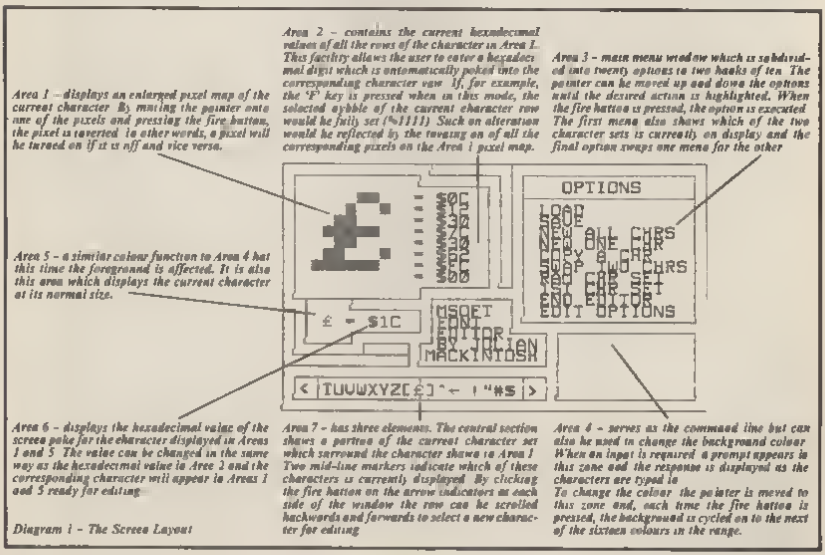


Diagram 1 – The Screen Layout

NEW ONE — *The character corresponding to the current RAM character is transferred from ROM*

COPY CHR — *Replaces the current character with the one whose screen poke hexadecimal value is specified in the input window*

SWAP CHR — *Behaves in a similar way to COPY CHAR but the two character shapes are swapped*

ROM/RAM CHR SET — *The character set used to form the editor screen display is toggled between the ROM and the RAM sets*

1st/2nd CHR SET — *Toggles between the first and second character sets*

END EDITOR — *Returns control to the Basic mode from which the editor was entered*

OTHER OPTIONS — *Selects the second set of menu options*

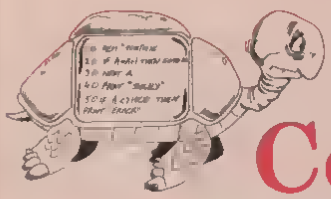
Table 2: The Manipulation Menu

COMMAND	FUNCTION
MIRROR UP	*Mirrors the bottom four lines of the character in the top four lines.*
MIRROR DOWN	*The reverse procedure of the MIRROR UP function*
MIRROR RIGHT	*Takes a mirror image of the four right-hand columns and copies them into the left four columns*
MIRROR LEFT	*Has the reverse effect of the MIRROR RIGHT function.*
FLIP VERT	*The current character is laterally reversed*
FLIP HORZ	*Inverts the current character*
ROTATE	*Rotates the current character through 90 degrees. Repeatedly select this option to rotate through 180 and 270 degrees*
INVERT	*Each pixel is reversed and so that any on pixels are turned off and vice versa.*
CLEAR	*The current character is erased*
MAIN MENU	*Selects the function menu*

```
PROGRAM  128 FONT EDITOR

 5 GRAPHIC1,1 REM MOVE @A5IC A0
 OVE M CODE
 6 GRAPHIC0,1
 10 DATA 3000
 20 DATA 4C 04 30,20,A5 20,8D,0
 3 30,20 62,36,A9,20,8D 03 30,2
 0 4E 36,052A
 30 DATA AD,2C,0A,29 F0 09,0A A
 D 2C,0A A2,00 8D,00 3C 9D,00 0
 4,8D,00 05C5
 40 DATA 3D 3D,00,05,8D,00 3E,5
 D 00,05 8D 00,3F,9D 00,07,3D,0
 0 3A 9D 05B1
 50 DATA 00 0C,8D,00,38 9D 00,0
 D E8 D0 D9,A9,00 20 D2 FF 20 0
 0 0C 4C 075C
 60 DATA 2F 30,AE,96,30,E8 E0,1
 0 D0 02 A2,00,EC,97,30,F0 F4 0
 E 96 30,0A12
 70 DATA 0A 4A A2,D0 86 FC A2,0
 0 86 FB A0 00,91 FB C8,D0,FB,E
 6 FC A6,0D3A
 80 DATA FC E0,DC,D0 F1 60,AA A
 D,05,30,20 D2,FF 60 90,05 1C,5
 F 9C 1E,0659
 90 DATA 1F,9E,81 55 96 97 96 5
 9 9A,98 01 02,AE 97,30,E0,E0 1
 0,D0,02,0506
 100 DATA A2 00,EC,96 30 F0,F4,
 0E 21 D0,0E,97 30 60 A2 C2 06,
 F8 A2,05 0AF3
 110 DATA 06 FC,A2 00 A0,00,A9
 20,91,FB C8 C0,00,D0 F9 A5 FC,
 18 69,20 0A0E
 120 DATA 85 FC A5,FC,69,00,09
 FC E8 E0,05 D0,E3 60 60 5D 05
 40 20,F0,003E
 130 DATA 30 0D,02,05 0E,63,06,
 00 40 20 10 31 60,10 69,F0,A0,
 00,59 4D 0594
 140 DATA 07,10,65,01,C8,C0,11
 D0 F5,60,10 40 20 05 31,AA,60
 4A 4A,4A,05E0
 150 DATA 4A,29,0F,C9 0A 90,02,
 E9 39 69 30,00 20 10,31 4C 31
 31 AE 03 05C0
 160 DATA 30 06,FC A2,00,00 FD
 A2,03 10 0A,20 FD,CA,D0 F9,85,
 F0,A5,FD 0070
 170 DATA 65 FC 85 FC,60,A2,04,
 0E 4F 31,A2,A4,0E 4E 31 A0 00
 A2,00 01,053C
 180 DATA F0,30,06 31,F0,05 A9
 A0 4C 4D 31,A9 20,90 E4,05,E0,
 E0,00 D0,03E0
 190 DATA EA,AD 4F,31,00 0E,31,
 AD 4E 31,60 60 31,01,F0,20 F0
 30,40 0A 0363
 200 DATA A2 00 48,66,90,0C 05,
 CA,E0 02 D0,F7,A5,20,18,60,4E,
 31 80,4E,00E9
 210 DATA 31,50,03,EE,4F,31,C0
 C0,09,D0 00,60,60,40,20,10,00,
 04,02,01,05A7
 220 DATA 48 0A,4E,AD 5D 06,20,
 16,31,50 10,65 F0 05,F0,90,02,
 E6,FC 60 00CD
 230 DATA AA,0D 8E 31 A2,00 41,
 F0 01,F8,AD,5D 06,4C,10,31 C9,
 0A,F0 20 00FA
 240 DATA 18 0A AA 00,00,50,C7,
 31,40,05 C6 31 40,60 F7 34 57,
 35 61 36,0875
 250 DATA 3F 36,05,37,52,37,A5,
 37 D0,37 20,30 A2 22,06,FD,A2,
 3E,00,FE,00H7
 260 DATA A2,DF,00,F8,A2,04,06,
 FC A5,0A,40,A0 00,01,F0,40,01,
 FD 01,F1 00F3
 270 DATA C8 91,FD,C8,C0,0E D0
 F1,A5,F8,10,03,20,A5,F0,90,02,
 E6 FC,A5,0C2F
 280 DATA FD,10,03,0E,05,FD,90,
 02,E6 FE,60,10,03,FF,D0,D2 AD,
 21,32,45,0A57
 290 DATA 01,0D,21 32 60 00,20,
 0D,05 12,12 0F,12,20,15,10 20,
 20 20 20 0281
 300 DATA 20 0D,05 12,12,0F 12
 20 04 0F,17,0E 20 20 20 0D,05,
 12 12,0F 017C
 310 DATA 12 20,12,09,07,00 14
 20,20,0D 05,12,12,0F,12,20 0C,
 05 00,14 0156
 320 DATA 20 20,20 05 0C,09,10
 20,00 0F,12,1A,20,20,20,20,20,
 05 0C 09 01A9
 330 DATA 10,20,16,05,12,14,20,
 20 20 20,20,12,0F,14,01,14,09,
 20,20 20,01C0
 340 DATA 20,20,20,20,20,03,0C,
 05,01,12,20,20 20,20,20,20,20,
 20 20,05,01F0
 350 DATA 0E,15,05,12,14,20,20,
 20 20 20,20,20 A0 0D 01,03,0E,
 A0,0F,00,0503
 360 DATA 54 05 0F,0E,53 A0 C9
 0A,D0,03,4C,D0 31,10,0A A0 00,
 00 05,C4,030F
 370 DATA 32,48,05,C3,32,48 E0,
 05 33 2C,33,0D 34,D6,33 53,33,
 44,34 03,0632
 380 DATA 33 D4,32,E0,32,AD,5D,
 05,20 10,31,A0,00,50,51,F0 C8
 C0 00,D0,00F1
 390 DATA F3,A0,5D,05,20 10,31,
 60 A0,5D 00,20,16 31 A0 00 01,
 F0 43,FF,07D5
 400 DATA 51,F8,C0 C8,00 D0 F5
 A0 5D,06,20 10,31,60 A0 5D 05
 20 16,31,0325
 410 DATA A0,04,A2,03,46 FD 01
 F8,04,FE,A4,FD,31,F0,E5,FE 88,
 04 FD A4 00D0
 420 DATA FE 98,C0 05,D0 EC A0
 5D 05 20 10 31 60 A0 5D,05,20,
 16,31,A2,080D
 430 DATA 04 A0,03 86,FD 01,F0,
 04,FE A4,FD,31 F0,C6,FE C8,04,
 FD,04,FE,0E34
 440 DATA 98 C0 FF D0,EC A0 5D,
 05 20 10 31,60 A0 5D,05 20 16,
 31 A0 00,0004
 450 DATA A2,07 05,FD,01 F0,48,
 04 FE A4,FD,01,F0,05,FD 60,01,
 F0,A5,FD 0E07
 460 DATA 04,FD,A4,FE,01,F0,C6,
 FD,C8 C0 04 D0,E3,A0 5D,00,20,
 10 31 60 0002
 470 DATA A0 5D,06,20,15,31,A2,
 00 0A 5D CF 33,E0,E8,08,D0,F0,
 A2 00,A0 091C
 480 DATA 00 01 F0,3D,00 31,F0,
 0A 05 C7 33 10 7D CF,33,9D,CF,
 33 C8 C0 0ADD
 490 DATA 00 D0 EA E0 E0 00 D0
 E3 A0 00 05,CF 33 91,F0,C8,C0,
 00 D0 FB 0C02
 500 DATA AD,5D,06 20 10,31,60
 01 02 04 00 10 20,40,00 00 10,
 00 E3,0F 03D2
 510 DATA 11 31,C7,A0,5D,06,20,
 10 31,A0 00 01 F0,29 F0 91,F8,
 C8 C0 05 0301
 520 DATA D0,F5,A0,00 A2,00,01
 F0,3D,00,31,F0,00 0D C7 33,10,
 71,F0 91 0A6D
 530 DATA F8,E8,E0 04 D0 EC C8,
 C0,00,D0 E5 A0,5D,06,20 10,31,
```

```
60,AD,9D 0AA3
540 DATA 06,20,10 31 A4 00 01
F0 25 0F 51 F0,C0,C0,00,D0,F5,
A0,00 A2 0814
550 DATA B4 01 F0 3D 00 31,F0
00 0D C7,33 10 71 F0 91,F0,E0,
E0 00 D0 0885
560 DATA EC C0 C0,00,D0,E5,AD,
9D 00,20,10 31 00,A2,00,0A,9D,
CF,33 E8 0985
570 DATA E0 00,D0,F0,AD,9D,00,
20 10 31 A0 00,A2 00,01,F0,3D,
00,31 F0 09F0
580 DATA 0A,05 CF,33,10,7D,C7,
33 55 CF,33,E6,E0 00 D0 EA,C6,
C0,05 D0 0A05
590 DATA E3 A0 00,05,CF,33,01,
F0,C0,C0 00 D0,F6 AD,5D,00,20,
10 31,00 09F1
600 DATA 10 A2,11 A0 1A 20 F0
FF,00 A2 00 A0 20 9D EC 34,E0,
E0,00,D0,050F
610 DATA F0 A2 00 06,F0,20,E4,
FF F0 F0 40 A5 07 20,D2,FF 60,
C9 0D F0,0C20
620 DATA 1A,C9,14,F0,17 C9,23
90,E6 C9,70,00 E4,A4 F0,E6,F0,
98 EC 34 0C73
630 DATA 20,D2 FF,C8,0A,90 D6
60 A4 F6,F0 D1 C8,F0,A9,9D 20,
D2 FF,A9 0082
640 DATA 20 20 D2 FF A9 91 20,
D2 FF A9,20 00,99,EC,34 4C,A1
34 45 2E 05E7
650 DATA 20 20 20 20 20 20,20,
20 20 20 20 20 20 60 34 A2 00
0D AE 35,0951
660 DATA 20,D2,FF,EE,EF,17 D0,
F5,18 A2 13 A0,1A 20 F0 FF 20
91,34,A9 0483
670 DATA 60,20 99,FF,A0,0D,A2,
EA A0,34 20 0D FF A9 00,A2,00,
A0,00,20,0884
680 DATA 0A FF A9,00,A2,00,AC,
03 30,20 D5,FF,00,07,AD,5D,00,
20 10 31,07FF
690 DATA 60 10 A2,15 A0 1D 20,
F0 FF A2 00,00,53 35 20 D2,FF
E0,E0,05,03A0
700 DATA D0 F5 00 45 52 52 4F,
52,20 00 34 A2,00 0D,C0,35,20,
D2,FF,E6,050E
710 DATA E0 17 D0 F5 A2 13,A0,
1A 18 20 F0,FF 20 91,34 A9 00
20 30 FF 058F
720 DATA A5 0D A2,EA A0 34,20,
0D FF A9,00 A2 00 A0,01,20,0A,
FF,AE 03 0570
730 DATA 30,00,FC,A2 00 00,F0
A9,F0,A2,00,AC,03,30,C0,C9 C8,
C0 C8,C8 00AA
740 DATA C0,C0,20,0D,FF,00,96,
AD,5D,09 20,10,31,00 20,20,20,
4C 4F 41,07DA
750 DATA 44 11 9D 9D 9D,9D,9D,
9D,48 4F 4E,54 20 4E 41,4D,45,
20 20,20,0B0D
760 DATA 20 53 41,50,45 11,9D
9D,9D,9D 9D 48,4F,4E,54,20
4E 41 4D 0741
770 DATA 45,20,20,20,20,43,4F,
50 55 11 9D 9D 9D 9D,9D,43,
48 41,52 05DD
780 DATA 41 43,54,45,52,20,11,
9D,9D,9D,9D 9D 9D,9D,9D,9D,9D,
4E 55,4D,0882
790 DATA 42,45 52 20,24,20 20,
20,4D,4F,56,45 11 9D 9D 9D 9D,
9D,9D,43,06B5
800 DATA 48 41,52,41,43,54,45
52 20,11,9D,9D,9D,9D,9D,9D,9D,
9D,9D,9D,0950
810 DATA 4E,6S 4D,42,4S 52,20,
24,11,20,20,45 4E,54,45,52,20
41,9D,9D 0977
820 DATA 5D,9D,9D,9D 9D,9D,9D,
11,48,45,50,49,44,45,43,49 4D,
41,4C 9D,0016
830 DATA 9D 9D,9D,9D,9D,9D,9D,
11 4E 55 4D,42,45,52,AD,00,FF,
48,A9,01 00C3
840 DATA 8D 00,FF,A9,00,85,FB,
AD 03 30 10 65,B4,85 FC,A9 00
85,FD,AD 0A1F
850 DATA 03,36,05,FE,10,85,06,
AA,A0,00,01,F0,01,FD,C8,D0,F8,
E6,FC E6 0C1C
860 DATA FE E4,FE,D0,EF,60,8D,
00,FF AD,5D 06,20,10 31,60,AD,
5D,06,48,050C
870 DATA 20,16,31,A5,FC,65,00
85,FE A5 F0 85 FD A0 00 FF,40
A5,01,8D,0AF1
880 DATA 00,FF,A6,00 01,FD,01,
F8 C9,C9,00,D0,F7 68,8D 00,FF,
60 20,10,0ABC
890 DATA 31 60 20 AE,30 20,00,
34,A0,00,85,34,36,20 D2 FF C8
C0 2E,D0 00A5
900 DATA F5,20 E4,FF F0,F8 A2,
00 00,F5,36,F0,00,E8 E0 10 D0
F6,4C E1,0D51
910 DATA 36,60 30,31,32 33,34
35 36 37,38 39,41,42,43,44,45,
46,20 60,04E0
920 DATA 34,A2,00,8D,0D,2E,35,20
D2,FF E0 E0,20,D0 F5 20 E1,36,
20,D2,FF 0877
930 DATA 0A 0A 0A,0A,0A,85,FD,
20,E1,36,20,D2,FF 0A 85 FD 20,
16,31,A5,07F4
940 DATA F0,48,A5,FC,49 A0,5D,
06,20,16,31,60,65,FE,60,65,FD,
A0,00 01,05C9
950 DATA FD 51 F0,C0,C0,00,D0,
F7 AD,5D 00 20 10,31,00,20,06,
34,A2 00 05EF
960 DATA 0D 05 36,20,D2 FF E0,
E6 20 00 F5 20 E1 36,20,D2,FF,
0A 0A 0A,0A00
970 DATA 0A 0A,05,FD,20,E1 36
20 D2,FF,0A,05 FD 20 10 31 A5
F0 48,A5 053E
980 DATA FC,48,AD,5D,00 20,16,
31 60 05 FE 60,05,FD,A0 00 01,
F0 48-01-05D5
990 DATA FD,01,F0 00 51 FD,C8,
C0 00,D0 F1 AD,5D,00,20,10,31,
00 AD 2C 0A7A
1000 DATA 0A,20 0C C9,04 D0 15
,AD,2C,0A,29 F3,05,00,8D,2C,0A
,AD,D1,05 0847
1010 DATA 29,00,05,01,8D,D1,05
,00,AD 2C 0A,29,F3,00,04,0D,2C
,0A AD,D1 06C3
1020 DATA 05 29,00,00,0F,8D,D1
,05 60 AD 2C,0A,29,02,F0,23,A9
,0E,20 D2 0603
1030 DATA FF A2,00,BD,F9,05,29
,00,10,26,38,90,F8,05 E8 E8 03
,D0 F0,A9 04AD
1040 DATA 20,0D,03 30,AD,5D,06
 20,10,31,60 A5,0E,20 D2,FF,A2
,00 0D F8,0780
1050 DATA 05,29,00 1D,29 30,9D
 F8 05 E8 E0 03 D0 F0 A9 20 0D
,03,30,AD,000F
1060 DATA 5D 06 20,10 31 60 31
,13,14,32,0E 04 60 60 60 AE 00
,0D,F0,F0,05FC
1070 DATA AD,65,0D,C9,FF F0 10
 40 A0 00 0C 66 0D 20 AE 30,05
,AE,21,32,0040
1080 DATA D0,09,20,04,31,4C,2F
,38 20,AE,32,4C,2F,30,AD,03,0D
 AC 04 0D,0570
1090 DATA C9 04,90 22,C9,0C,00
 1E C0,04,90,1A,C0 0C,00 16,00
,00 00 00,0042
1100 DATA 04,F0,A6,F8,18,69,FC
,20,90 31 A0 00 0C,00 0D 4C 2F
,30,C9 04 085D
1110 DATA 90,17,C9,07,D0,13,C0
 0E,90 0F C0 11,00,00,20,52,30
,A0,00 0C 0701
1120 DATA 66 0D,4C,2F,30,C9,1A
 90 17 C9 25 00 13,C0,11,D0,0F
,C9 16,00,0757
1130 DATA 00,20,90 30,A0,00,0C
,00 0D,4C,2F,30,C9,02 90,1D,C9
,05 00 19,0654
1140 DATA C0,14,90 15,C0,17 00
,11,EE 5D,06,AD,5D,06,20,D6,30
,A0,00,0C,07C4
1150 DATA 66,0D,4C,2F,30,C9,16
,90 1D,C9 15 00 19 C9 14 90,15
,C0,17 00,0750
1160 DATA 11 CE 5D 06 AD 5D 06
,20,D6,30,A0,00 0C,66,0D,4C,2F
 30,C9 10 06A3
1170 DATA 90 23,C9 12 00 1F C0
,04,90,18,C0 0C,00 17 18 69 F8
,90 00 05,0668
1180 DATA 06,05,F8 58 A6,F8,20
,42,39 A5 00 0D 66 0D 4C 2F 30
 C9 0A,90,0030
1190 DATA 1A,C9 0C,00 16 C9 0F
,90 12,C9 10,00,0E,18 69,F6 20
,7D,38,A0,07A1
1200 DATA 00,0D,66,0D,4C,2F,30
,4C 2F 38,40,EA 40,20,CE,30 00
,FE AD 5D 0732
1210 DATA 06,20,10,31,00,D0,19
,60 A0,A5 FE,0A,0A,0A,0A,05,FE
,01,F0 23 07F1
1220 DATA 0F 05,FE,51,F0,AD,5D
,05,20,D5,30,00 60 A0 01,F0,25
,F0 05 FE 0A0C
1230 DATA 51 F0 AD 5D 00 20,D0
,30,60 40,20,CE 30 05 FE,A5,5D
 05 F0 A5 0A41
1240 DATA 06,05 FC,00 AA A9 00
,40 E0 00 D0 D0 4C,53,35,00 00
,00,00,00,02EA
1250 DATA 00 00 00,00,00 00 00
,00 00,00 00,00 00 00 00 00 00
,00 00 00 0000
1260 DATA 00,00,00 00 00,00,00
,00 00,00 00,00 00,00,00 00 00
,00,00 00,0000
1270 DATA 00 05,00 00,00,00,00
,00,00 00,00,00,00,00,00,00 00
 00 00 0000
1280 DATA 00,00,00,00 00,00,00
,00,00,00,00 00 00,00,00,00,00
```

```
,00,00,00,0000
1250 DATA 00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1300 DATA A2 48,CA,BD,67,0D,9D
 00,8E,E0,00,D0,F5,A0,04,0A,2S
,FE,BD,04 003C
1310 DATA 0A,AD,15,D0,29,FE,85
,01,AD,15 D0,A9,5E BD,00,D0,BD
,01,D0,AD 00AE
1320 DATA 18,D0,29,FE,8D,18,D0
,A9,30,8D F0,07,A9 F1,8D,27,D0
,AD,14,03 03C3
1330 DATA 9D 5F,0D,AD,15,03,0D
,00 0D,70,A9 52,8D,14,03,A9,0C
,8D 15 03 0325
1340 DATA 50 00,AD,05,0D,C9,FF
,F0,03 20 30,0D,AD,20,DC,AA,20
,01,D0 05 002D
1350 DATA A9 FC,10,6D,01,D0,8D
,01,D0,8A 29 02,D0,09,A5,04 10
 0D,01 D0,07EA
1360 DATA 0D,01,D0,0A,29,04,D0
 13 A5,FC,10 0D,00,D0,0D,00,D0
,00,20 AD 0084
1370 DATA 10,D0,49,01,8D,10,D0
,8A 29,00,D0 13 A5 04,10,0D,00
,D0 0D 00 05C4
1380 DATA D0 00,8A,AD,10,D0 49
,01 0D 10 D0,0A,18,AD,00,D0,4A
,AA AD,10,05AC
1390 DATA D0,29,01,F0,05 0A 10
,05,8E AA 8A,4A,4A,10,09,F0,8D
,01,0D AD 0008
1400 DATA 01 D0 4A 4A,10,09,01
 4A,10 09 F5,0D,E2,0D,98 20,10
,D0,4E,AE,8714
1410 DATA 66 0D D0 10,AD,01,0D
,CD 63,0D D0,08,AD,62,0D,CD,64
 0D F0 11,07DE
1420 DATA AD,01,0D,0D,63,0D,AD
,62 0D,0D,64,0D A9,01,0D,66,0D
,AD,61,0D 09F7
1430 DATA C9,25,00 19,C9 17 30
,15,AD 62,0D 18,69,F8,C9,0A,00
,06 69 01,07CC
1440 DATA 0D,55,0D,20 30 0D,4C
,5C 0D A5,FF,0D,85,0D,4C,5C,0D
,A2,00,8E 05A5
1450 DATA 03,0D,A2,00 0C,04 0D
 4C 01,0D,EA,EA,A2,0C,04,02 A0
,04 04 03,30A2
1460 DATA AA A9 28,18,65 A2,50
,02,E6,03,05,02,CA,D0 F2,A0,0E
,A5,90,19,0A97
1470 DATA 71 02 01,02 A0 D0,F0
,00,0C,5F,0D,00 00,00,00 00
,FF 00,C0,07A0
1480 DATA 00 00,E0,00,00,E0,00
,00 70 10,00,70 FE,20 70,EF,00
,3F,77,00,0500
1490 DATA 30 01 00,30 0D,C0 1D
 0D C0 1C 70,E0 1D,F7 E0,00,EF
,F0,0F,DF,0AD3
1500 DATA F0 07 1F F0,07,CF,F0
 03,EF,F0,01,F7,E0,00,FF,C0,00
 7F,00,00 09D4
1510 DATA 1C,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00 00 00,001C
1520 DATA 00,00,00,00,00,00,00
,00 00 00 00 00,00,00,00,00,00
,00,00 00 0000
1530 DATA 00,00,00,20,20 00,00
 00,00,00 00,20,00 00 00 00 00
,00,00 00,0000
1540 DATA 00,00,00 00 00 00 00
 00,00,00,00,00,00,00,00,00,00
,00,00 00,0000
1550 DATA 00 00 00,00,00,00 00
,00,00,00,00,00,20,20,20,20,20
,20,20,20,0100
1560 DATA 20,20,20,20,20,20,20
,20,20,20,20 20 20 20,20,20,20
,20,20,20,0200
1570 DATA 20,20,20,20,20,20,20
,20,20,20,70,40,40,40 40,40,40
,40,40,40,0390
1580 DATA 40,40,40,40 40 40,72
,40 40,40,40,40,40,72,40,40,40
,40,40 40,0564
1590 DATA 40,40,40,40,40,40 40
,40,40,40,GE,20,20,5D,70,40,40
 40 40,40,0530
1600 DATA 40 40,40,40,40 GE,GD
,40 40,40,40,40,GE,5D,70,40,40
,40,40,40,0506
1610 DATA 40,40,40,40 40,40,40
,40,40 0E,5D 20 20 5D,5D,20,20
,20,20,20 04A5
1620 DATA 20,20,20,20,20,GD,40
,40,40 40 40,0E,5D,5D,5D,20,20
,20,0F,10 0451
1630 DATA 14,09,0F,0E,13,20,20
,20,20 5D,5D,20,20,5D,5D,20,20
,20,20,20 0321
1640 DATA 20 20 20,20,20 3D 20
,24 30 30 20,5D,5D,5D,60,40,40
 40,40 40,0463
1650 DATA 40,40,40,40 40,40,40
,40,40,73,5D,20,20,5D,5D,20,20
,20 20 20 04AA
1660 DATA 20 20,20,20,20,30,20
,24,30 30,20,5D,5D,5D,20,0C
,0F,01 04 0355
1670 DATA 20,20,20,20,20,20,20
,20,20,5D,5D,20 5D,5D,20,20
,20,20,20,0374
1680 DATA 20 20 20 20,20,3D,20
,24 30,30 20,5D,5D,5D,5D 20,13
,01,15 05 0354
1690 DATA 20 20,20 20,20,20,20
,20 20 5D 5D,20 20,5D,5D,20,20
,20 20 20,0374
1700 DATA 20 20,20,20,20,3D,20
,24 30 30 20,5D 5D,5D 5D,20,0E
 05 17,20,037F
1710 DATA 01,0C,0C,20 03 00,12
,13 20,5D,5D 20 20 5D,5D 20 20
,20,20 20 02DD
1720 DATA 20 20 20 20,20,3D 20
,24,30,3D 20 5D,5D 5D 5D 20,0E
,05,17 20 037F
1730 DATA 0F 0E,05,20,03 00,12
,20 20 5D 5D,20,20 5D 5D 20 20
,20,20 20 02F3
1740 DATA 20 20 20 20 20 3D,20
,24,30,30 20 5D,5D,5D 5D,20,03
,0F,10 15,0370
1750 DATA 20,01,20,03 20,12,20
,20 20 5D,5D,20,20,5D,5D,20,20
,20 20 20 0312
1760 DATA 20 20 20,20,20 3D 20
 24,30,30,20 5D,5D,6D,5D,20 13
,17 01 10,0370
1770 DATA 20 14,17,0F,20,03,00
 12,13 5D 5D 20,20,5D,5D,20,20
 20 20 20 02FE
1780 DATA 20,20,20,20,20,3D,20
 24 30,30 20,5D,5D,5D,20 12
,01 0D 20,0375
1790 DATA 03,00,12 20,13 05 14
,20 20 5D,5D,20 20 5D,5D,20,20
,20 20 20,02FD
1800 DATA 20,20,20,20,20,70,40
,40 40,40 40 7D 6D,7D,6D,20,31
,13,14,20,04AC
1810 DATA 03 00,12,20,13 05,14
,20 20,5D 5D,20,20,5D 5D,72,40
,40,40,72,0411
1820 DATA 40,40,40,40 40,70,70
,40,40,40,40 40 6E,5D,20,05
,0E,04,20 04CF
1830 DATA 05 04,05,14,0F,12,20
,20,20,5D,5D,20,20 6E,6E,5D,20
,20,20 5D,03A4
1840 DATA 40,40,40,40,40,6E,5D
,0D,13 0F 0E 14,20 5D,5D,20 05
,04,05,14,0374
1850 DATA 20 0F 10,14,05,0F,0E
 13 6E 5D,5D 20,20 5D,5D 5D,20
,20 20 3D,035A
1860 DATA 20 24,32 30,20,5D,5D
,05 0F 0E,14,20,20,5D,0D,40,40
,40,40 40,0401
1870 DATA 40,40,40,40,40,40,40
,40,40 7D 5D,20,20,5D,5D,5D,20
,20,20,70,0541
1880 DATA 40,40,40,40 40,7D,5D
,05,04,05 14,0F,12,6D,40,40,0E
,70,40 40,04AC
1890 DATA 40,40,40,40 40 40,40
,40,40,6E,5D 20,20,5D 5D 6D 40
,40 40 7D 05AF
1900 DATA 70 40 40,40,6E,70,7D
,02,15 20,0A,15 0C 05 01,0E,5D
,5D 20,20 0403
1910 DATA 20,20,20,20,20,20,20
,20,20 5D 5D 20 20,5D,6D,40,40
,40,40,40,0424
1920 DATA 71 40,40 40,73 5D 0D
,01,03,05,05,0E,14,0F,13,0E,5D
,5D,20 03 034F
1930 DATA 31,32 30 20,04,05,13
,03,20,5D,5D,20,20,6D,40,40,40
,40 40 40 03E3
1940 DATA 40 40,40,40,7D,6D,40
 40,40,40,40 40,40,40,40,40,7D
 5D,20,20,05B4
1950 DATA 1E,05,12,13 05 0F 0E
,20,20,5D,5D,20,20,5D,70,40,72
,40,40,40,03DF
1960 DATA 40,40,40,40,40,72,40
,40,40 40,40,40 40,72,40,6E
,5D,20,20,0577
1970 DATA 03,20,20,20,31,35,30
,37,20,5D,5D,20,20,5D 5D,3C,5D
,10,15,1A 02F1
1980 DATA 1B 1C 1D,1E 1F,20 21
,22,23,24 25 26,27,28,5D,3E,5D
,5D,20,20 036A
1990 DATA 20 20 20 20,20,20,20
 20,20 5D,5D 20,20,5D,6D,40,71
 40,40 40 0455
2000 DATA 40,40,40 40 40,71,40
,40 40,40 40,40 40,71,40 7D
,0D 40 40 05CC
2010 DATA 40 40,40,40 40 40 40
,40 40 7D 5D 20 20,0D,40,40,40
,40,40 40 0547
2020 DATA 40,40,40,40,40 40 40
,40 40,40,40,40,40 40 40,40,40
,40,40,40 0580
2030 DATA 40,40,40,40 40,40,40
 40,40,40,7D 20 20,20 20,20,20
,20 20 20 041D
2040 DATA 20,20,20 20,20,20,20
,20 20,20 20,20,20,20,20,20,20
,20,20 20 0280
2050 DATA 20,20,20,20 20,20,20
,20,20,20 20 20 00 20 00 00 00
 00,00 00 0108
2060 DATA END
@3995 PRINT"DATA ERROR IN LINE
"PEEK(65)+256*PEEK(66) END
@3995 LOOP INPUT"FILENAME FOR
SAVE  ";N$ @SAVE(N$),08,P(5)TOP
(E) END
```



GTX Compiler

Supercharge your Basic programs by converting them to pure machine code

The GTX Compiler is designed to convert a Commodore 64 program written in Basic into high speed, machine code This code makes calls to standard functions stored in memory, called the run-time library which means that the compiled program is shorter because frequently-used routines such as add or subtract can be stored as subroutines

Before using GTX, the program to be compiled must already be saved on tape or disk. When the compiler runs, the title page will appear and three prompts for inputs given in sequence

The SOURCE NAME is the name of the Basic program to be compiled If using tape, press RETURN to load the program

The OBJECT NAME is the name that the compiled program will be saved under If using tape, RETURN will cause the program to be saved without a name

SIGN SPACE refers to the printing of numbers In Basic a number is preceded by a sign space and ended with a space Pressing RETURN will print numbers in this way Altering the Y to N will print numbers with neither of the two spaces.

The program specified by the source name will then be loaded and compiling will begin There are two parts to this process

PASS 1
Each command is deciphered and the corresponding machine code is produced The current Basic program line number is printed along with the present length of the compiled program but, if any errors occur, the compiler stops and the error message is printed The Basic program must then be re-loaded and modified if it is to be compiled successfully

Apart from errors, any number of warnings may be given These do not stop the compiler since the program can still be compiled but the results obtained with the original, Basic version will differ from those in the compiled version Again, the original program must be modified if the results are to tally

PASS 2
All jumps such as GOTO and GOSUB have the correct addresses inserted and any DATA is transferred within the program

If compiling succeeds, then the Basic program length and the compiled program length are printed along with the following options

R Run the compiled program
S Save the compiled program
O Output the run-time library
C Compile another program
E exit

It is advised that the program is saved before running (option S) in case there is a problem If using tape, a copy of the run-time library must be saved directly after the compiled program (option O) If using disk, a single copy of the run-time library will serve all the programs on the disk Finally, the compiled program can be run using option R

Acceptable Basic

The compiler does not accept the full set of commands available in Basic Those that can be compiled are

PRINT (including , and ,)
POKE
PEEK
IF/THEN
READ/DATA/RESTORE
FOR/NEXT
GOSUB/RETURN
GOTO
CLR
SYS
REM
AND
OR
END
STOP

There are also restrictions placed on these commands but, although all this seems rather daunting, most programs require very little attention before they can be successfully compiled

All numbers are 16 bit un-signed integers
This obviously means that GTX can not be used for complex programs

by R D Bell

that require floating point numbers but the main concept for GTX is to transform relatively simple Basic programs, such as games, to fast machine code programs This type of program usually only requires integer numbers so this limitation is not a major one. The lack of negative numbers can cause problems in converting a program that was not specifically written to be compiled However, this limitation can normally be solved by changing the style of programming to incorporate numbers between 0 and 65535

For example, part of a normal Basic program consists of the following.

```
10 PRINT"[CLS]"
20 P=10*40+1024+20:D=1
30 A=P:P=P+D
40 IFP>=10*40+1024+39
   THEND=-D
50 IFP<=10*40+1024
   THEND=-D
60 FORB=1TO500:NEXT
70 POKEP,88:POKEA,32:
   GOTO30
```

As it stands this would not compile because of the negation of D in lines 40 and 50 The possible values for D are 1 and −1, the latter being out of range on GTX. Therefore, before the program could be compiled, a few simple modifications would need to be made

Lines 30 to 50 must be changed to the following.

```
30 A=P:P=P-1:IFD=1
   THENP=P+2
40 IFP>=10*40+1024+39
   THEND=1-D
50 IFP<=10*40+1024
   THEND=1-D
```

Now the program could be compiled In fact, when it was compiled it ran 36 times faster than the original

Variables can only be single letters
This can cause problems if the original has a large number of variables To assist with this problem there is a single array available, Z, which is pre-dimensioned to 64 elements ie Z(0) to Z(63) The DIM in the original program is not required by GTX but if included it will be skipped Any further dimensioned arrays will cause an error to be flagged

Expressions are evaluated from left to right
In Basic expressions are calculated in a set order, not from left to right Brackets take top priority, next come indices, followed by division and multiplication then addition and subtraction, with Boolean expressions (AND, OR, NOT) taking lowest priority For example,

```
A=3+6*2
```

In Basic this expression would evaluate A as being equal to 15 6*2=12 and 12+3=15 If compiled A would be calculated as being equal to 18 3+6=9 and 9*2=18 To ensure the compiled program and the Basic program obtain the same value the expression would need to be changed to

```
A=6*2+3
```

This can cause problems when a program was written without compiling in mind By checking the order of all expressions and changing this where necessary, the compiled program should obtain the same results as the Basic version During Pass 1 of compiling the order of expressions are checked and, if discrepancies are likely to occur a warning is given The indicated expressions should be re-ordered to ensure the program will operate correctly

Brackets are not valid expressions
Brackets are only allowed with a PEEK command Any other brackets must be removed before the program can be compiled Usually a number of dummy variables are required to achieve this An expression such as

```
A=5+((3*X)-8)/7-T/(4*Y-2)
```

would have to be changed to

```
A=3*X:8/7+5:B=4*Y-2:B=T/B
A=A-B
```

Only one PEEK per expression
An expression such as this

```
A=PEEK(45)+PEEK(46)*256
(order problem anyway)
```

would have to be changed to

```
A=PEEK(46)*256:A=A+PEEK(45)
```

before it could be compiled by GTX This means the original Basic program is not as neat and will run a little slower but, thanks to the speed increase given by GTX, this is insignificant when compiled

No strings
Strings are not incorporated at all. The use of strings in the type of programs intended to be compiled is usually very infrequent Their only use is usually connected with a GET There are a number of methods of emulating GET with the commands available, this is discussed later

AND and OR can not be used in IF/THEN comparisons
A program line which takes the form

```
10 IFA=3ORA=6THEN 1000
```

would have to be split into two separate lines before it could be compiled

```
10 IFA=3THEN1000
11 IFA=6THEN1000
```

The AND statement must also be changed when used in the context.

```
10 IFA=4ANDA=7THEN1000
```

This would become

```
10 IFA=4THENIFA=7THEN1000
```

AND and OR are included in the list of available commands but are only legal when acting as arithmetic operators Take the situation where it is necessary to strip off the five most significant bits from the contents of location 197

```
B=PEEK(197)AND7
```

In this case, the AND is not operating as a logic comparator so the expression can be compiled If AND or OR are used in an IF/THEN statement their operation would still be treated as arithmetical and the Basic program would not operate in the same way as the compiled version

FOR/NEXT loops can not use STEP
The STEP function can be emulated by adding another statement into the FOR/NEXT loop

```
10 FORA=0TO940STEP40 POKE
   1024+A,102 NEXT
```

would have to be changed to

```
10 FORA=0TO940 POKE1024+
   A,102 A=A+39 NEXT
```

before it could be compiled Note that A is only increased by 39 because the NEXT statement automatically increments it by 1, resulting in a step of 40 for each loop

If the STEP is negative, it creates more of a problem

```
10 FORA=20TO0STEP-2
   PRINTA NEXT
```

would have to be changed to

```
10 FORB=0TO20:A=20-B
   :PRINTA B=B+1:NEXT
```

Either of the above cases could be solved by a conditional GOTO loop Taking another example

```
10 FOREA=920TO40STEP-40
   POKE1024+A,160 NEXT
```

could be changed to:

```
 9 A=920
10 POKE1024+A,160 A=A-40
   IFA>=40THEN10
```

READ/DATA
This is almost the same as Basic except that the numbers stored in the DATA statements can only be eight bits (0 to 255 decimal) This is not a serious limitation in games because most DATA statements are used for storing user-defined characters or sprite data, where the values are only eight bits anyway.

RESTORE is exactly the same as in Basic

The Z( ) array is restricted
This array must always be used through another variable

```
10 POKE Z(A),44
```

must he changed to:

```
10 B=Z(A) POKEB 44
```

Missing commands

The missing commands are rarely used and can often be forgotten or emulated using the legal commands

GET can be emulated in two ways The first, and simplest, involves reading the keyboard directly using a PEEK. The values returned for each key are non-standard but for inputting only a few keys this is not a problem The second method allows the keys to be read in ASCII and actually uses the same Kernal ROM routine as GET

```
10 GETA$ IFA$="A"THEN100
```

can be changed to either:

```
10 IFPEEK(197)=10THEN100
```

or

```
10 SYS65508 IFPEEK(780)=
   65THEN100
```

In the first case, a value is returned for as long as the key is held down unlike the second case where the value is returned only once for each key press

RND can be emulated by reading certain memory locations that change frequently Two of the best locations to use are the raster register (53266) and the CIA timer A (56324)

```
10 A=INT(RND(1)*10)
```

would have to be changed to:

```
10 A=PEEK(56324)AND15
   IFA>9THEN10
```

A different type of random statement would be:

```
10 IFRND(1)< 1THENPRINT
   "HELLO"
```

Such a statement has a 10% probability of printing 'HELLO' The best way to achieve this with the compiler is to change it to

```
10 IFPEEK(56324)<25THEN
   PRINT"HELLO"
```

This also gives a 10% probability because there are 256 possible values for location 56324 The statement only reacts to 25 of these values, so the probability calculation becomes (25/256)*100, which equals 10

CHR$ statements can usually be changed to a direct PRINT If it is not possible to do this in your program then this alternative may be used

```
10 PRINT CHR$(A*4-9)
```

can be emulated by

```
10 POKE780,A*4-9 SYS65490
   PRINT
```

The extra PRINT is required because the character is printed without a carriage return

Typing it in

If you are using a disk then the programs can be entered and saved in any order With a tape they must be saved in the correct order

First, enter Program 1, the loader that changes the address at which the compiler is loaded GTX must always be loaded at this new location or it will be over-written by the program to be compiled Save this loader at the start of the tape

Next enter Program 2 the actual compiler Do not run it yet because it must be loaded at the new address It can be typed in at its run-location by entering the following direct command before starting.

```
POKE44,100 POKE43,1 POKE
25600,0 NEW
```

When the program has been entered, save it using the filename "GTX COMPILER", following on directly from Program 1

Ensure that the computer has been reset before typing in Program 3, the run-time library Before running, save this program on a spare tape or disk in case of fatal errors Run the program and insert the original tape or disk taking care not to erase the other two programs The run-time library will be saved with the filename "RTL". The run-time library must always be saved with this name even when working with a cassette tape

You should now have a complete Basic compiler

Testing the compiler

The compiler is now ready to be tested Enter Program 4, the test program, and save it Load and run GTX and compile the program If

any errors are detected, correct your test program and try again Run the compiled program using option 'R' and you should see GTX COMPILER filling the screen

If you run the program in Basic first, you will notice the speed increase, typically 45 times Pressing 'R' will re-start and 'E' will exit back to Basic

Memory requirements

The compiler uses the following locations which should not be altered by the compiled program If they are, crashes will inevitably occur

$C000–$C2FF (49152–49919) These locations contain the compilers run-time library This must always be present whenever the compiler itself or a compiled program is used In both cases, it will be loaded automatically by the program If it can't be found, the program will crash

$CF00–$CFFF (52992–53247) The compiled program's variables are stored here This location can be changed, if required, by altering the value of VA in line 11 of the compiler

A compiled program also uses a few page zero locations which, again, must not be altered by the compiled program–

$02 (2)
$14–$15 (20–21)
$39–$3C (57–60)
$3F–$42 (63–66)
$FB–$FE (251–254)

Program Description

Line numbers

30–890	Prints current line number and length Deciphers next command and jumps to corresponding part of the program
1000–2510	Evaluates expression
3000–3070	GOTO and GOSUB
3200–3930	IF/THEN
4000–4600	PRINT
5000–5030	POKE
5100–5160	FOR
5200 5260	NEXT
5300–5310	REM
5400–5420	DIM
5500–5570	SYS
5600–5670	DATA
5700–5770	READ
8000–8030	Prints 'insert tape', or disk, depending on current device and then waits for a key to be pressed
9000–9070	Opening title page Inputs source name, object name and sign space
9100–9120	Loads program specified by the source name from the current device to $0801 (2049 decimal)
9500–9505	Relocates program to run at $0820 (2080 decimal)
9510–9570	Prints original length and compiled length along with the total number of warnings The options are also printed
9580–9650	Inputs selected option
9700–9720	Checks if run-time library is present, if not it is loaded off the current device to $C000 (49152 decimal)
9800–9830	Saves the compiled program to current device
10000–10190	Inserts the correct addresses for all GOTOs and GOSUBs
10200–10220	Copies all DATA from its temporary store starting at $C700 (50944 decimal), to its correct place in the compiled program
20000–20550	Copies the machine code out of the run-time library to its correct place in the compiled program
25000–26050	Simplifies the machine code, if possible
30000–32000	Error messages.

Variables

PR	Pointer to the next byte of the Basic program
L%( )	Line numbers of the Basic program
A%( )	Start address of the machine code for each Basic line
J%( )	Address of each jump, such aa GOTO and GOSUB
T%( )	Address of each THEN
N%	Number of Basic lines encountered so far
CM%	Current Basic token being processed
VA	Start address of compiled program's variables
JP	Number of GOTOs and GOSUBs so far.
TH	Number of THENs encountered so far
DD	Start of temporary store for DATA ($C700, 50944 decimal)
EN	End address of program being compiled
DA	Start address of machine code
AD	Address to store next byte of machine code
AJ	Adjust value to relocate program to $0820 (2080 decimal)
DV	Current I/O device (1=tape and 8=disk)

PROGRAM BTX COMPILER - LOADER

```
5E   10 PRINT"{CLR}{DOWN3}{F}OKE44,I
     00 POKE43,1 POKE2560 0 NEW"

37   20 PRINT"{DOWN2}LOAD"CHR$(34
     )"BTX COMPILER"CHR$(34)" "PE
     EK(186)

60   30 PRINT"{DOWN3}RUN{HOME}"
CF   40 FORA=0TO5 POKE631+A,13:NE
     XT POKE198,10
02   50 END
```

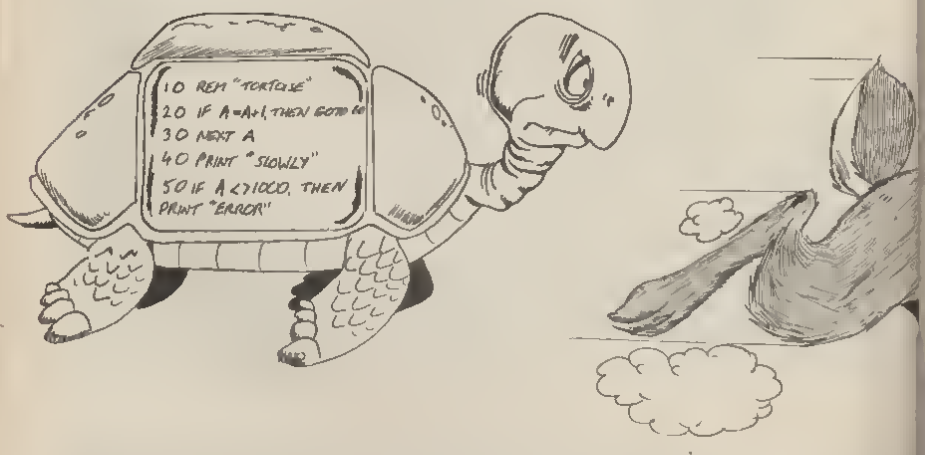
PROGRAM BTX COMPILER - MAIN

```
16   1 GOTO9700
A3   10 PR=2049 DIML%(300) A%(300
     ),J%(300),T%(300) N%(29)
1F   11 N%=-1 UA=52992 JP=0 TH=0
     DO=50944 WAS="{HOME DOWN2}"
FI   15 PRINT"{CLR,5F,5A 552} 1"
F0   20 EN=PEEK(173)*256+PEEK(172
     ):AD=EN+4 IFA0<211ORENA0=211
     0
F5   25 OA=A0 AJ=A0-2107 GOSUB205
     50 GOSUB20500
0A   30 LN=PEEK(PR+3)*256+PEEK(PR
     +2) PR=PR+4
CA   31 IFPR>=ENTHEN10000
49   33 IFA0>25500THEN30210
50   35 PRINT"{HOME DOWN} LN,A0 O
     A
65   40 N%=N%+1 L%(N%)=LN A%(N%)=
     A0-AJ
10   50 CM%=PEEK(PR)
FE   60 IFCM%<153THEN4000
A3   70 IFCM%>64ANDCM%<91THEN2000

26   80 IFCM%=137ORCM%=141THEN300
     0
D5   90 IFCM%=128ORCM%=142ORCM%=1
     44THENGOSUB20270 Z%=0 GOTO90
     8
0F   100 IFCM%=139THEN3200
AF   110 IFCM%=151THEN5000
53   120 IFCM%=129THEN5100
04   130 IFCM%=130THEN5200
D4   140 IFCM%=156THENGOSUB20500
     Z%=R GOTO900
07   150 IFCM%=143THEN5300
74   160 IFCM%=134THEN5400
78   170 IFCM%=158THEN5500
51   180 IFCM%=131THEN5600
F3   190 IFCM%=148THENGOSUB20540
     Z%=0 OOTO9R0
50   200 IFCM%=135THEN5700
4E   210 IFCM%=136THENPR=PR+I GOT
     O50
A0   050 GOTO30200
25   500 PR=PR+2%+I
04   510 IFPEEK(PR)=58THENPR=PR+1
     GOTO50
07   520 IFPEEK(PR)=0THENPR=PR+1
     GOTO30
C9   530 GOTO30150
14   1000 Z%=0:WA%=5
2C   1010 Z5%=PEEK(PR+2%) IF29%=0
     OR29%=50OR29%=44OR29%=164THE
     N1020
44   1011 IF29%=55OR29%=34OR29%=1
     78OR29%=179OR29%=177OR29%=16
     7ITHEN1020
08   1013 Z%=Z%+1 OOTO1010
40   1020 Z%=Z%-1 A%=0 B%=0 C%=0
     FORA=0TO2%
35   1030 IFPEEK(PR+A)=154ANDA%=1
     THEN30000
4C   1040 IFPEEK(PR+A)=154ANDPEEK
     (PR+A+1)<>40THEN30010
55   1050 IFPEEK(PR+A)=154THENA%=
     1 C%=I GOTO1100
3C   1060 IFPEEK(PR+A)=40ANDB%=1T
     HEN30020
41   1070 IFPEEK(PR+A)=40THENB%=1
     OOTO1100
00   1000 IFPEEK(PR+A)=41ANDB%=0T
     HEN30030
70   1090 IFPEEK(PR+A)=41THENB%=0
     A%=0
D2   1100 NEXT
7C   1110 IFC%=ITHENI150
FE   1120 X%=0:Y%=Z% O4%=0 GOSUB2
     100
94   1130 GOSUB20120
F6   1140 RETURN
QE   1150 O0%=0 FORG=0TO2% IFPEEK
     (PR+0)=154THENO6%=O6%+1
10   1160 NEXT X%=2 Y%=X%
73   1170 IFPEEK(PR+Y%)<>41THENY%
     -Y%+1 GOTOI170
85   1100 Y%-Y%-1 O1%-0 GOSUB2100
     GOSUB20130
64   1150 GOSUB20150 X%-Y%+2 IFO6
     %=1THEN1260
B4   1200 GOSUB20120 O4%=1 A%=PEE
     K(PR+X%) GOSUB2230 O2%=O1%
16   1210 IFPEEK(PR+X%+1)<>154THE
     N30000
D2   1220 X%=X%+3 Y%=X%
C6   1230 IFPEEK(PR+Y%)<>41THENY%
     -Y%+1 GOTO1230
EC   1240 Y%-Y%-1 O1%-0 GOSUB2100
     GOSUB20130
22   1250 GOSUB20140 O1%=O2% GOSU
     B2500 X%=Y%+2
47   1260 O4%=2 O1%=0 Y%=Z% IFX%>
     =Y%THEN1200
0D   1270 OOSUB2100 O1%=0
11   1200 GOSUB20120 RETURN
A2   2000 PR=PR+1
C7   2001 IFPEEK(PR)=40ANDCM%<>80
     THEN30150
91   2002 IFPEEK(PR)=40THENOO=PR+
     1 GOTO2050
A6   2005 OO=0 IFPEEK(PR)<>170THE
     N30040
07   2010 PR=PR+I IFPEEK(PR+I)=40
     ANDPEEK(PR)<>50THEN2015
62   2011 IFPEEK(PR+1)=40THENGOSU
     B2050 OO=1 GOTO2019
07   2015 O1%=0 GOSUB1000
E5   2015 O=UA+2+(CM%-65) H%=O/25
     6 L%=O-H%*256
IF   2020 BOSUB20160
31   2030 IFOO=1THEN2056
CF   2040 GOTO500
CE   2050 IFPEEK(PR)<>170THENPR=P
     R+I GOTO2050
D6   2051 PR=PR+1
12   2092 IFPEEK(PR+I)=40ANDPEEK(
     PR)=50THEN30150
00   2053 OI%=0 GOSUB1000
A6   2054 BOSUB20310
24   2055 PR-OO-2:GOSUB2050
01   2056 IFPEEK(PR)=0ORPEEK(PR)=
     50THEN910
C0   2057 PR=PR+I:GOTO2050
A5   2000 X%=0 Y%=0
36   2061 IFPEEK(PR+Y%)<>41THENY%
     =Y%+1 BOTO2001
14   2062 Y%-Y%-1 RETURN
E0   2090 PR-PR+2:O1%=0:GOSUB2000
     GOSUB2100
4F   2051 IFOO=0THENGOSUB20120:GO
     SUB20310 GOSUB20400 RETURN
EA   2092 GOSUB20450 RETURN
5O   2100 A$="  O%=0 FORA=X%TOY%+
     1 A%=PEEK(PR+A)
20   2101 IFO%=ZTHENO4%=0 GOTO22
     30
CF   2110 IFA%>47ANDA%<58ANDO%=1T
     HEN30050
04   211I IFA%>47ANDA%<58THENAR=A
     0+CHR0(A%):00TO2300
B%   2120 IFAR=""THENZI60
0C   2130 O=VAL(A$) X%=O/256 L%=O
     -H%*256 AR=""
CF   2131 IFO1%<>0THEN2151
90   2140 GOSUB20000
2F   2150 O%=1 GOTO2230
90   2151 BOSUB20040
07   2152 O%=1 GOSU02500:BOTO2230

AF   2160 1FA%>64ANDA%<91ANDO%=1T
     HEN30050
01   2170 IFA%>64ANDA%<91THEN2190

07   2180 GOTO2220
4A   2190 O=VA+2*(A%-55) H%=O/256
     L%=O-H%*256
FA   2191 IFO1%<>0THEN2211
DD   2200 GOSU020020
C1   2210 O%=1 OOTO2300
0E   2211 GOSU020050
09   2212 O%=1 GOSU02500 GOTO2300

AC   2220 IFO%=0THEN30050
12   2230 IFA%=170THENO1%=3 O%=0
     BOTO2300
B4   2240 IFA%=172THENO1%=5 O%=0
     BOTO2300
F0   2250 IFA%=171THENO1%=4 O%=0
     BOTO2300
07   2251 IFA%=173THENO1%=6 O%=0
     BOTO2300
69   2252 IFA%=175THENO1%=1 O%=0
     BOTO2300
69   2200 IFA%=176THENO1%=2 O%=0
     BOTO2300
C0   2261 IFA%=0ORA%=50ORA%=44ORA
     %=41ORA%=59ORA%=34THEN2300
AC   2202 IFA%=170ORA%=175ORA%=17
     7ORA%=167ORA%=164THEN2300
24   2270 GOTO30060
37   2300 IFO%=ITHENO4%=0 RETURN
40   2310 NEXT RETURN
C3   256R ONOI%BOSUB20460,20470,2
     0000,20100,20050,20110
AC   2510 IFWA%<INT((O1%+1)/2)THE
     NWA=WA+1:WA=WA+"{DOWN}":PR
     INTWA"{5L}1NE"LN"WARNING"
E4   2520 WA%=INT((O1%+1)/2) RETU
     RN
00   3000 PR=PR+1 A$=" FORZ=0TO6
     A%=PEEK(PR+Z)
5F   3010 IFA%=58ORA%=0THEN3040
2C   3020 IFA%>47ANDA%<58THENA$=A
```

```
    0+CHR@(A%) NEXT
1C  3030 GOTO30070
E0  3040 O=VAL(A$) H%=O/256 L%=O
    -H%*256 Z%=Z-1
00  3050 IFCM%-1=1ITHEN3070
F1  3060 GOSUB20230 GOTO580
05  3070 GOSUB20250 GOTO580
I0  3200 PR=PR+1 IFPEEK(PR)=107T
    HEN30056
20  3201 OI%=0
C2  3210 GOSUBI000 PR=PR+2%+1 A%
    =PEEK(PR)!A1%=PEEK(PR+1)
F2  3220 CF%=0 IFA%=167THEN3700
90  3230 PR=PR+2:IF(A%=179ANDA1%
    =177)OR(A%=177ANDA1%=179)THE
    NCF%=1 GOTO3300
0D  3240 IF(A%=177ANDA1%=178)OR(
    A%=178ANDA1%=177)THENCF%=2:G
    OTO3300
4A  3256 IF(A%=179ANDA1%=178)OR(
    A%=178ANDA1%=179)THENCF%=3 G
    OTO3300
AF  3255 IF(A1%<64ORA1%>90)AND(A
    I%<47ORA1%>58)ANDA1%<>194THE
    N30120
F4  3260 PR=PR-I:IFA%=177THENCF%
    =4:GOTO3300
26  3270 IFA%=178THENCF%=5 GOTO3
    300
F6  3280 IFA%=179THENCF%=6 GOTO3
    300
22  3290 GOTO3011@
13  3300 GOSUB20310
0C  3310 O1%=0 GOSUBI000 IFPEEK(
    PR+2%+I)<>167THEN30120
F0  3320 ONCF%GOSUB20330 20410,2
    0450,20440 20320,20420
01  3330 GOTO3500
C4  3700 GOSUB20300
FI  3710 Z%=-1 GOTO3800
CF  3800 T%(TX)=A0+1 TX=TX+1
19  3810 GOSUB20280
00  3840 GOTO3900
F0  3500 PR=PR+2%+1 IFPEEK(PR)<>
    167THEN30120
94  3510 IFPEEK(PR+1)>47ANDPEEK(
    PR+1)<58THENCM%=137 GOTO3000

2A  3520 PR=PR+1
25  3530 GOTO50
05  4000 Z1%=0
40  4010 Z1%=Z1%+1
I4  4020 A%=PEEK(PR+Z1%)
40  4030 IFA%=3%ITHEN4000
E0  4050 IFA%>0!ANDA%<51THEN4200

7F  4051 IFA%>47ANDA%<58THEN4200

A9  4052 IFA%=194THEN4200
AD  4050 GOSUB4500 IFA=ITHEN4400

EA  4001 GOTO4010
OI  4000 Z1%=Z1%+1 Y1%=Z1%
CC  4030 A%=PEEK(PR+Z1%) IFA%=0T
    XEN4110
9E  4100 IFA%<>34THENZ1%=Z1%+1 G
    OTO4050
4E  4110 GOSUB20100
0C  4120 IFZ1%-Y1%=0THEN4101
6E  4130 FORA=0TOZ1%-Y1%-1!POKEA
    O PEEK(PR+Y1%+A) AO=AO+1 NEX
    T

B4  4101 IFA%=0THENGOSUB4500 GOT
    O4400
E0  4170 Z1%=Z1%+1 A%=PEEK(PR+Z1
    %) GOSUB4500 IFA=1THEN4400
72  4100 GOTO4010
E9  4200 R1=PR PR=PR+Z1%
F4  4205 O4%=0 O1%=0 O2%=0 O5%=0
    GOSUB10000
C2  4210 GOSUB20210
E7  4220 PR=R1 Z1%-Z1%+2% A%=PEE
    K(PR+Z1%)
6C  4230 GOSUB4500 IFA=1THEN4400

A4  4231 IFA%=34THEN21%=Z1%-1
AE  4240 GOTO4010
D0  4300 Z%=Z1%-1 OOTO500
29  4500 A=0 IFA%<59ORA%>34THENR
    ETURN
E9  4510 IFA%=44THEN4550
3C  4520 IFA%>64ANDA%<91THENRETU
    RN
CD  4530 IFA%=194THENRETURN
50  4531 IFA%=0ORA%=58THEN4570
43  4540 RETURN
0E  4550 GOSUB20200
6F  4560 RETURN
23  4570 A=I IFPEEK(PR+Z1%-1)=59
    ORPEEK(PR+Z1%-1)=44THENRETUR
    N
2E  4500 GOSUB20220
07  4600 RETURN
3E  5000 PR=PR+1 O1%=0 GOSUB1000
    GOSUB20310
3F  5010 PR=PR+2%+2 O1%=0 GOSUB1
    000
03  5020 GOSUB20340
05  5030 GOTO500
74  5100 PR=PR+1 CV%=PEEK(PR) IF
    CV%<65ORCV%>90THEN30130
E2  5110 PR=PR+1 IFPEEK(PR)<>170
    THEN30130
54  5120 PR=PR+1 O1%=0 GOSUB1000
     PR=PR+2%+2 IFPEEK(PR-1)<>10
    4THEN30130
CB  5130 Q=VA+2*(CV%-65)!H%=O/25
    6 L%=O-H%*256 GOSUB20160 CM=
    Q+52
B5  5140 O1%=0 GOSUB10000
70  5150 M%=CM/250 L%=CM-M%*250
    GOSUB20160
1C  5160 NX(CV%-65)=AO-AJ NX%(NX
    )=CV% NX=NX+1 GOTO500
CA  5200 PR=PR+1 A%=PEEK(PR) IFA
    %=32THEN5200
60  5201 IFA%=0ORA%=58THEN2%=-1
    NX=NX-1 A%=NX%(NX) GOTO5220
EF  5210 Z%=0 IFA%<65ORA%>30THEN
    30140
03  5211 NX=NX-1 IFA%<>NX%(NX)THE
    N30230
05  5220 A%=A%-65:Q=VA+2*A% H%=Q
    /256 L%=O-H%*250
0A  5230 Q=Q+2*20 M1%=O/256:L1%=
    O-M1%*255
33  5240 GOSUB20350
45  5250 PR=PR+2%+1 IFPEEK(PR)=3
    4THEN5200
03  5260 GOTO510
79  5300 IFPEEK(PR)<>0THENPR=PR+
    1 GOTO5300
D5  5310 OOTO510
0E  5400 PR=PR+1 IFPEEK(PR)<>30T
    HEN30160
3F  5410 IFPEEK(PR)<>0ANDPEEK(PR
    )<>58THENPR=PR+1:GOTO5410
60  5420 OOTO510
27  5500 PR=PR+1 O1%=0 GOSUB1000

70  5560 GOSUB20510
07  5570 OOTO500
35  5600 A$=" ":PR=PR+1
0E  5610 A%=PEEK(PR)!IFA%<48ORA%
    >57THEN5630
71  5620 A@=A@+CHR@(A%):PR=PR+1:
    GOTO5610
F4  5630 Q=VAL(A@) IFQ>255THEN30
    170
EF  5640 POKE00,Q 00=00+I
67  5650 IFA%=44THEN5600
69  5660 IFA%=58ORA%=0THEN510
61  5670 OOTO30100
DC  5700 PR=PR+1 A%=PEEK(PR)
19  5710 IFA%<65ORA%>90THEN30190
02  5720 Q=VA+2*(A% 65)!H%=O/256
    L%=O-H%*256
AE  5730 GOSUB20520
04  5740 PR=PR+1 A%=PEEK(PR)
7A  5750 IFA%=44THEN5700
FF  5760 IFA%=0ORA%=58THEN310
04  5770 GOTO30150
F0  6000 PRINT"CDOWN,9PCS)  IFO
    V=1THENPRINT"POSITION TAPE".
    GOTO6020
0F  6010 PRINT" INSERT DISK"
30  6020 PRINT" AND PRESS A KEY(
    DOWN)" POKE198,0 WAITI98,1 P
    OKE198,0 RETURN
46  5000 POKE53280,14 POKE53281,
    6 POKE198,0
7C  5005 PRINTCHR$(14)"CCLR,WHIT
    E,DOWN]  CSG,ST SX] CSC 50,5
    M,SP,SI,SL,SE,SR,SPC(8)C(SC)
    ) CSR] CSD] CSB]ELL"
1C  5010 PRINT"  CCYI2)"
D4  5020 INPUT CDOWN2,RIGHT2,SS)
    OURCE NAME  NS$
D6  5021 IFLEN(NS$)=0ANDOU<>1THE
    NPRINT"CUP3)"  GOTO5020
60  5030 INPUT"CRIGHT2 SO)OJECT
    NAME",NO@
0F  5031 IFLEN(NO$)=0ANDOU<>1THE
    NPRINT"CUP3)"  GOTO5030
05  5040 IFOU<>1ANONS@=NO@THENPR
    INT"CUP,RIGHT14,SPC28,UP3" @
    OTO5030
32  5050 INPUT"CDOWN RIGHT2 SSJI
    GN-SPACE? YCLEFT3J";A$
0E  5070 SP%=0 IFA@="Y"THENSP%=1

70  5100 GOSUB@@@@:POKE10S,@:T=4
    0960-LEN(N5@) H%=T/256 L%=T-
    H%*256
0D  5110 POKE702,H% POKE701,L% P
    OKE700,LEN(N5@) SYS65469
1A  5120 POKE701 I POKE702 @ POK
    E700,@ SYS65493 OOTO10
E9  5500 H%=OA/256 L%=OA-H%*250
    POKE251,L% POKE252 H%
42  5505 H%=AO/256 L%=AO-H%*255:
    POKE253,L% POKE254,H% SY5497
    40
40  0510 POKE53200,14 POKE53201,
    0
9C  5911 PRINT"CCLR,WHITE,DOWN,R
```

```
   IGHT3,C@16)" PRINT"CRIGHT3 R
   VSON,SP,SR,SO S@,SR,SA,SM,SS
   PC 5C,SO,SM SP,SI,SL,SE,SD,R
   VSOFF)"
01  90I2 PRINT'CDOWN,RIGHT3,S@,S
   A 55 5I SC) LENGTH "EN=2049,
   TA@(26) "2049",-EN"CLEFT3)"
9C  9613 PRINI CRIGHI3 SCJDMPILE
   D LENGTH  AD-DA+S@ TA@(26)"C
   2049" -(2107+AD-DA)"CLEFTJ)"
2A  9514 PRINT CDOWN,RIGHT3,SW)A
   RNING9 "UN
ER  9520 PRINI"CDOWN2,RIGHI3,S5)
   ELECT ONE OF THE FOLLOWING -
   '
40  9530 PRINI"CDOWN,RIGHT6,RVSO
   N,S9,RVSOFF)  SAVE COMPILED
   PROGRAM"
30  9540 PRINI'CDOWN,RIGHT6,RVSD
   N,SR,RVSOFF)  RUN COMPILED P
   ROGRAM AND EXII
05  9550 PRINI CDOWN RIGHT@,RVSO
   N,SD,RVSOFF)  OUTPUT RUN-TIM
   E LIBRARY"
G4  9560 PRINT"CDOWN,RIGHT6,RVSD
   N,SC RVSOFF)  COMPILE ANOTHE
   R PROGRAM"
28  9570 PRINI'CDOWN RIGHT6,RVSO
   N,SE,RVSOFF)  RETURN TO @ASI
   C"
9A  9580 POKE158,@
46  9590 GETA$ IFA$=""R THENSY520
   80 END
57  96I0 IFA$="C"INENRUN
5D  9620 IFA$="S"THENGOSUB3800 G
   OIO9510
CE  9630 IFA$="O"INENGOSUB8000 S
   YS49854 GOIO9510
C5  9640 IFA$="E"IHENEND
E1  9650 GOIO9500
EE  9700 POKE1020,82 POKE1021,84
     POKEI022,76 DU=PEEK(I6G) IF
   PEEK(49192)=16DIHEN3000
10  9710 POKEI05 @ POKE782,3 POK
   E781,252 POKE780,3 SYS65469
E8  9720 POKE781 @ POKE782,152 P
   OKE780,@ SYS65493 GOTO3000
19  9800 GOSUB8000 I=49960 LEN(N
   S$) LEN(NO$) X%=I/256 L%=I-H
   %*256
70  9810 POKE782 X% POKE781,L% P
   OKE780 LEN(NO$) SYS65469
43  9820 POKE254,@ POKE253,1 POK
   E780 253 A=AD-AJ H%=A/256 L%
   =A-H%*256
55  9830 POKE782,X% POKE781,L% S
   YS65496 RETURN
FD  10000 PRINI CHOME,DOWN,SP,SA
     @S2) 2(SPC7)"
34  10001 A%(N%+1)=AD-AJ GOSUB20
   270
DD  10010 IFJP-@IHEN10100
43  10020 FORA=@IOJP-1
69  10030 A%=PEEK(J%(A))+256*PEE
   K(J%(A)+1)
0F  10040 FO@@=@ION% IFL%(@)<>A%
   THENNEXI LN=A% GOIO3@@@
D8  10050 H%=A%(@)/256 L%=A%(@)-
   256*H%
@A  10070 POKEJ%(A) L% POKEJ%(A)
   +1 H%
@6  10000 FORB=@IO@ NEXT
13  10050 NEXT
26  10100 IFIH=@IHEN10200
SI  10120 FORA=@IOIH-1
59  10130 A%=PEEK(I%(A))+256*PEE
   K(I%(A)+1)
@5  10140 FOR@=@ION% IFL%(@)<>A%
   THENNEXI LN=A% GOTO3@@@@
55  10150 @=@+1 H%=A%(@)/256 L%=
   A%(@)-256*H%
FE  10170 POKEI%(A),L%:POKEI%(A)
   +1,H%
A2  10180 FOR@=@IO@ NEXI
77  10190 NEXT
@5  10200 IFOO=50944IHEN9500
E3  10210 FORA=@IOOO-5@5N5 POKEA
   @+A,PEEK(50544+A) NEXT @=AD-
   AJ
@4  10220 H%=@/256 L%=@-H%*256 P
   OKEDA+3,H% POKEDA+1,L% AD=AD
   +A GOIO95@@
@A  20000 FORV=@IO7 POKEAD+V,PEE
   K(49152+V) NEXI
53  20010 POKEAD=1,L% POKEAD+5,H
   % AD=AD+V CI=1 GOTO25@@@
7F  20020 FORV=@ID5 POKEAD+V,PEE
   K(49168+V) NEXI
87  20030 POKEAD=I,L% POKEAD+2,H
   % POKEAD+6,L%+I POKEAD+7,H%
   AD=AD+V CI=2 GOTO25@@@
2C  20040 FORV=@ID7 POKEAD+V,PEE
   K(49178+V) NEXT
25  20050 POKEAD+1 L% POKEAD+5,H
   % AD=AD+V CI=3 GOTO25@@@
7E  20060 FORV=@I05 POKEAD+V,PEE
   K(49178+V) NEXI
19  20070 POKEAD+1,L% POKEAD+2,H
   % POKEAD+6,L%+1 POKEAD+7,H%
   AD=AD+V CI=4 GOTO25@@@
C3  20080 FORV=@IO2 POKEAD+V,PEE
   K(49186+V) NEXI AD=AD+V CI=5
     RETURN
52  20090 FORV=@IO2 POKEAD+V,PEE
   K(49191+V) NEXI AD=AD+V CI=6
     RETURN
52  20100 FORV=@IO2 POKEAD+V PEE
   K(49194+V) NEXI AD=AD+V CI=7
     RETURN
9E  20110 FORV=@IO2 POKEAD+V,PEE
   K(49197+V) NEXI AD=AD+V CI=8
     RETURN
67  20120 FORV=@IO7 POKEAD+V,PEE
   K(49200+V) NEXI AD=AD+V CI=9
     GOIO25@@@
D@  20130 FORV=@ID7 POKEAD+V PEE
   K(49208+V) NEXI AD=AD+V CI=I
   @ RETURN
@5  20140 FORV=@ID7 POKEAD+V,PEE
   K(49216+V) NEXI AD=AD+V CI=1
   1 GOTO25@@@
@7  20150 FORV=@IO7 POKEAD+V,PEE
   K(49224+V) NEXI AD=AD+V CI=1
   2 GOTO25@@@
CF  20160 FORV=@ID9 POKEAD+V,PEE
   K(49232+V) NEXT CI=13
@1  20170 POKEAD+3,L% POKEAD+4,H
   % POKEAD+8 L%+1:POKEAD+9,H%:
   AD=AD+V GOTO25@@@
A@  20180 FORV=@IO6:POKEAD+V,PEE
   K(49242+V) NEXT
44  20190 POKEAD+1,Z1%-Y1%:AD=AD
   +V CI=14 RETURN
@D  20200 FORV=@IO2 POKEAD+V,PEE
   K(49245+V) NEXI:AD=AD+V CI=1
   5 RETURN
57  20210 IFSP%=1THENOO5U820430
@A  20211 FORV=@IO6 POKEAD+V PEE
   K(49252+V) NEXI;AD=AD+V IF5P
   %=1THENOOSU820430
D7  20212 CI=16 RETURN
DD  20220 FORV=@ID4:POKEAD+V PEE
   K(49259+V) NEXT AD=AD+V CI=1
   7 RETURN
D4  20230 FORV=@IO2:POKEAD+V,PEE
   K(49264+V) NEXI:J%(JP)=AD+1
A@  20240 POKEAD+I L% POKEAD+2,H
   % AD=AD+V JP=JP+1 CI=18 RETU
   RN
4F  20250 FORV=@IO2 POKEAD+V PEE
   K(49267+V) NEXI J%(JP)=AD+I
C5  20260 POKEAD+1,L%:POKEAD+2,H
   % AD=AD+V JP=JP+1 CI=19 REIU
   RN
9D  20270 POKEAD,PEEK(49270) AD=
   AD+I CI=20 RETURN
32  20280 FORV=@IO2 POKEAD+V PEE
   K(49271+V) NEXT
E3  20290 POKEAD+2,LN/256:POKEAD
   +1,LN-PEEK(AD+2)*256 AD=AD+V
     CI=21 RETURN
26  20300 FORV=@IO7 POKEAD+V,PEE
   K(49274+V) NEXI AD=AD+V CI=2
   2 RETURN
A@  20310 FORV=@IO7 POKEAD+V,PEE
   K(49282+V) NEXT AD=AD+V CI=2
   3 GOTO25@@@
18  20320 FORV=@IO11 POKEAD+V,PE
   EK(49288+V) NEXI AD=AD+V CI=
   24 RETURN
16  20330 FORV=@IO11 POKEAD+V,PE
   EK(49300+V) NEXT AD=AD+V CI=
   25 RETURN
70  20340 FORV=@IO5 POKEAD+V,PEE
   K(49314+V) NEXI AD=AD+V CI=2
   6 RETURN
F3  20350 FORV=@IO25 POKEAD+V PE
   EK(49328+V) NEXT CI=27
1F  20360 POKEAD+1,L% POKEAD+2,H
   % POKEAD+5,L%+1 POKEAD+10,H%
7A  20370 POKEAD+4 L1% POKEAD+5
   H1% POKEAD+12,LI%+1 POKEAD+1
   3,M1%
BD  20380 POKEAD+17,L% POKEAD+18
     H% POKEAD+20,L%:POKEAD+21 H
   %
83  20390 POKEAD+25,L%+1 POKEAD+
   25 H%
D8  20400 POKEAD+29,N%(A%)/256 P
   OKEAD+28 N%(A%)-PEEK(AD+28)*
   256 AD=AD+V RETURN
78  20410 FORV=@IO15 POKEAD+V PE
   EK(49350+V) NEXI AD=AD+V CI=
   28 RETURN
AE  20420 FORV=@IO15 POKEAD+V,PE
   EK(49366+V) NEXT AD=AD+V:CI=
   29 RETURN
90  20430 FORV=@IO4 POKEAD+V,PEE
   K(49382+V) NEXT:AD=AD+V:CI=3
   0 RETURN
88  20440 FORV=@IO17 POKEAD+V,PE
   EK(49387+V) NEXT AD=AD+V:CI=
   31 RETURN
18  20450 FORV=@IO17:POKEAD+V PE
   EK(49405+V) NEXT:AD=AD+V:CI=
   32 RETURN
```

```
71  20480 FORV=0TO2 POKEAD+V,PEE
    K(49423+V) NEXT AD=AD+U CI=3
    3 RETURN
45  20490 FORV=0TO2 POKEAD+V,PEE
    K(49428+V) NEXT AD=AD+U CI=3
    4 RETURN
F4  20500 FORV=0TO2 POKEAD+V PEE
    K(49428+V) NEXT AD=AD+U:CI=3
    5:RETURN
EF  20510 FORV=0TO2 POKEAD+V,PEE
    K(49432+V) NEXT AD=AD+U:CI=3
    6:RETURN
13  20520 FORV=0TO2 POKEAD+V,PEE
    K(49435+V) NEXT AD=AD+U CI=3
    7:RETURN
99  20530 FORV=0TO11 POKEAD+V,PE
    EK(49430+V) NEXT AD=AD+U CI=
    38 RETURN
A2  20540 FORV=0TO8 POKEAD+V,PEE
    K(49458+V) NEXT
AE  20550 POKEAD+1,L% POKEAD+3,H
    % AD=AD+U CI=39 RETURN
15  20560 FORV=0TO2 POKEAD+V,PEE
    K(49457+V) NEXT AD=AD+U CI=4
    0 RETURN
32  20570 FORV=0TO11 POKEAD+V,PE
    EK(49480+V):NEXT AD=AD+U CI=
    41 RETURN
2E  25000 IFLI=1ANDCI=9THENAD=AD
    -8 POKEAD-5 28 POKEAD-1,21 L
    I=42 RETURN
79  25010 IFLI=2ANDCI=9THENAD=AD
    -8 POKEAD 6,28 POKEAD-1,21 L
    I=43 RETURN
AA  25020 IFLI=9ANDCI=13THEN2600
    0
41  25030 IFLI=9ANDCI=23THENAD=A
    D-8:POKEAD-5,03:POKEAD-1,04
    LI=45 RETURN
03  25040 IFLI=11ANDCI=44THEN200
    20
60  25050 IFLI=11ANDCI=45THENAD=
    AD-8 POKEAD-5,63 POKEAD-1 64
    LI=47 RETURN
00  25060 IFLI=12ANDCI=44THEN200
    40
D9  25070 IFLI=12ANDCI=45THENAD=
    AD-8 POKEAD-5,03 POKEAD-1,64
    LI=49 RETURN
56  25500 LI=CI RETURN
5F  26000 AD=AD-18 FORV=0TO8 POK
    EAD+V PEEK(AD+8+V) NEXT
DO  26010 POKEAD+1,251 POKEAD+8,
    252 AD=AD+18 LI=44 RETURN
70  26020 AD=AD-18 FORV=0TO9 POK
    EAD+V PEEK(AD+8+V) NEXT
63  26030 POKEAD+1,20 POKEAD+6,2
    1 AD=AD+18 LI=48 RETURN
B7  26040 AD=AD-18 FORV=0TO9 POK
    EAD+V,PEEK(AD+8+V) NEXT
41  26050 POKEAD+1,253 POKEAD+6,
    254 AD=AD+18 LI=48 RETURN
26  30000 PRINT"{ST}OO MANY {SP,
    SE2 SK}S", GOTO32000
27  30010 PRINT"{SP,SE2,SK} WITH
    OUT {RACKETS", GOTO32000
58  30020 PRINT"{SR}RACKETS TOO
    COMPLEX", GOTO32000
D9  30030 PRINT"{SD}RACKET NESTI
    NG ERROR", GOTO32000
9F  30040 PRINT"{SL,SE,ST} WITH
    NO EQUAL", GOTO32000
74  30050 PRINT"{SE}XPECTING OPE
    RATOR", GOTO32000
96  30000 PRINT"{SI}LLEGAL OPERA
    ND", GOTO32000
20  30070 PRINT"{SG,SO SI SO}/{S
    G,SO SS SU SO} WITH NO NUMBE
    R", GOTO32000
E1  30080 PRINT"{SL}INE NUMBER D
    OESN'T EXIST" GOTO32000
2D  30090 PRINT"{SI SF}-{SI SH S
    E,SN} WITH NO EXPRESSION" G
    OTO32000
40  30100 PRINT"{SI SF} WITH NO
    {SI SH,SE SN}", GOTO32000
0A  30110 PRINT"{SC}OMPARISON ER
    ROR" GOTO32000
ED  30120 PRINT"{SI,SF}-{ST,SH S
    E,SN} GARBAGE", GOTO32000
78  30130 PRINT"{SF SD,SR}-{SN,S
    E,SX,ST} GARBAGE", GOTO32000

70  30140 PRINT"{SN,SE,SK,ST} WI
    THOUT {SF SO SR}", GOTO32000

05  30150 PRINT"{SA}RRAY GARBAGE
    ", GOTO32000
3A  30160 PRINT"{SD SI,SM} GARBA
    GE" GOTO32000
09  30170 PRINT {SN}UMBER TOO LA
    RGE" GOTO32000
4C  30180 PRINT"{SD SA,SI,SA} GA
    RBAGE" GOTO32000
EA  30190 PRINT"{SR,SE,SA SO} GA
    RBAGE ; GOTO32000
7A  30200 PRINT {SS}YNTAX ERROR
    ; GOTO32000
C4  30210 PRINT {SM}EMORY FULL -
     PROGRAM TOO LONG" END
33  30220 PRINT"{SF,SO SR}-{SN,S
    E,SX,ST} NESTING ERROR ! GOT
    O32000
E7  32000 PRINT" IN"LN END
```



Bus Route 64

Connect two Commodore 64 computers through their serial ports or link up to the C16 and Plus 4

There are many ways of connecting one Commodore computer to another but most methods require custom-made cables and complex software. This routine simply uses the serial bus which already has a cheap and easily available connector in the form of the disk drive/printer cable.

The serial bus has two lines which are capable of input and output on both computers: the CLock and DATa lines. There is also the ATN (attention) line that is used to interrupt external devices, but unfortunately it cannot accept a data signal. There is a line designated SRQ IN (serial request in) on the C64 but this has been deleted on the C16 and Plus/4 microcomputers.

The SRQ IN line allows external devices to interrupt the C64 and can only be used as an input. The CLK and DAT lines are so called because of their use in the Commodore Kernal ROM during communication with serial bus devices. It is the CLK line that governs which bits are valid on the DAT line and in a sense it 'clocks' the bits going out.

Recently a few dual-player games have appeared on the market where two Commodore computers are connected together and a game is loaded into both. This routine has possibilities in such environments, requiring very little modification to the actual code.

Two copies must be made to get the routine up and running, one per computer, with a slight modification at the beginning of the routine 'INT-VAR' in each version. Where it has LDA #XX, the XX must be 00 in one and 01 in the other. This is in order to condition TLKFLG (TaLK FLaG) so that one computer has the initial talk priority. This is toggled after every press of the return key.

```
                ;SERIAL BUS COMMUNICATIONS
                ORG   $0801
COLOUR          =$0286
                VIDEO INTERFACE CHIP #1 REGISTERS
VIC             =$9000
EXTCOL          =VIC+$20
BGCOLO          =VIC+$21
                COMPLEX INTERFACE ADAPTOR #2 REGISTERS
C12             =$FD00
C12PRA          =C12+$00
C12DRA          =C12+$02
                KERNAL ROM ROUTINES
CHROUT          =$FFD2
GETIN           =$FFE4
                BASIC HEADER
                WORD  EOB
                WORD  1987          LINE NUMBER
                BYTE  $9E           SYS TOKEN
                BYTE  2061
                BYTE  $00           SIGNALS END OF BASIC LINE
EOB             BYTE  $00,$00       SIGNALS END OF BASIC TEXT
SERCON          JSR INTVAR          INITIALISE VIDEO
                JSR LINREL          RELEASE SERIAL BUS LINES
                JSR INTVAR          INITIALISE VARIABLES
CHKFLG          LDA TALK
                BNE TALK            BRANCH TO 'TALK OR  LISTEN
                JMP LISTEN
TALK            JSR SETIA
                CMP #$00
                BEQ TALK            ;WAIT FOR CHAR FROM KEYBOARD
                STA BYTE
                PHA                 SAVE CHAR ON STACK
                LDX #$00            LIGHT GREEN TEXT
                STX COLOUR
                JSR CHROUT          ;PRINT CHARACTER TO SCREEN
                SEI                 ;THEN SEND ON SERIAL BUS
                JSR CLKLO           SEND ATTENTION
                JSR DATHI
WBATLO          JSR DETBIT
                BCS WBATLO          WAIT FOR ACKNOWLEDGE
                LDX #$00
                JSR CLKLO           MAKE DATA OUT VALID
NXTBIT          ROL BYTE            BIT TO SEND IN  C
                BCS HIBAT
                JSR DATLO           SEND ZERO BIT
                JMP BATWAI
HIBAT           JSR DATHI           ;SEND ONE BIT
BATWAI          JSR CLKHI           MAKE DATA VALID
                JSR DETBIT
WCLKLO          BNE WCLKLO
                JSR CLKHI
WCLKHI          BPL WCLKHI          WAIT FOR BIT ACKNOWLEDGE
                DEX
                BNE NXTBIT
                JSR DATHI           RELEASE DATA LINE
                CLI                 ALLOW MASKABLE INTERRUPTS
                PLA
                CMP #$00            IF  RETURN  KEY THEN LISTEN
                BNE CHKFLG
                DEC TLKFLG
```

When converting the routine to work with other Commodore computers which have a serial bus, the following table of register/bit numbers may be useful.

COMPUTER	CLK-IN	CLK-OUT	DAT-IN	DAT-OUT
C16, Plus/4	$0001/6	$0001/1	$0001/7	$0001/0
VIC 20	$911F/0	$912C/3	$911F/1	$912C/7
C64/128	$DD00/6	$DD00/4	$DD00/7	$DD00/5

by Simon Clarke

```
PROGRAM  BUS ROUTE 64

E3  10 GL=26      LN=50      SA=3814
    4
C0  20 FOR L=0 TO GL CX=0:FOR O=
    0 TO 15:READ A CX=CX+A
82  25 POKE53280,A:POKE SA+L*10+
    O,A:NEXT O
A5  30 READ A:IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
1D  40 NEXT L SY3804#0
5D  50 DATA 11 0,195 7,150,50,40
    ,54 40,0,0 0,32 113,0 32,785

ED  60 DATA 173 0,32 187,0 173,6
    53 6 208 3,76,181 0,32,220,2
    55,1753
EE  70 DATA 201,0,240,249 141,25
    2,0,72,162,13,142 134,2 32,2
    10 255 2113
11  80 DATA 120,32,145,0,32,163,
    0,176 251 162,0 32,145,0,14,
    252,1556
5F  90 DATA 0 176 6 32,127 0,76
    77 6 32 136 0 32 154,0 32,92
    0
62  100 DATA 163,0,40,251 32,163
    0 16,251 202,208,223,32,136
    0 96 1637
06  110 DATA 104 201 13 208 176
    206,253,0,240,171 169 1,141,
    253 0 96,2248
55  120 DATA 165 147,32,210,255
    165 0 141,32,208,141,33,208,
    58 173 0 2014
EA  130 DATA 221,5,32,141 0,221,
    56 173 0 221 41 223 141,0,22
    1,56,1836
CE  140 DATA 173 0,221,5 16 141,
    0 221,56,173,0 221,41,239 14
    1,0 1692
6A  150 DATA 221 56 173 6,221 20
    5,0 221 206,240,10,96 173 0,
    221 41 2134
3A  160 DATA 207 141 0 221,120,3
    2 163 0 40,251 32 127,0 32 2
    46 0 1644
96  170 DATA 36,136 0 162,0 32 1
    63 0 16 251,96 252 0,32,145
    6 1307
FC  180 DATA 36 240,0 32,154,0 3
    2 246,0 206,200,233 06,169,1
    0 141 1017
AB  190 DATA 134 2,173,252,0,32,
    616,255,173 252 0,201 13 208
    ,157 230 2356
A0  200 DATA 253,0,76 22,0 169 1
    20 136 208 253 96 0 0,0 0 0,
    1346
DF  210 DATA 100,0,133,250 169,1
    45,133 251 165,1 133,174,133
    ,183,169,0 2234
A5  220 DATA 133 175 133,154,169
    ,0 133,252,165 160,133,253,1
    60 6 177 250 2481
50  230 DATA 145 174 232,250 208
    ,2,230 251 630 174,250,2,230
    ,175,165 250,2824

48  240 DATA 187 252-200,234,165
    ,251 157 253 200 220 169,155
    ,133 187 165 150 3160
7F  250 DATA 133 100,169,12 133
    103 169,0,133 165,104 0,165,
    107,150 240,2147
01  260 DATA 6,32 210,255 200,20
    0,245,32,207 255,240,251,201
    ,45,240 4,2535
2E  270 DATA 201,56,40,230 41 15
    ,133,169 76,234,245 147 17 1
    7 73 70,1757

    300             BCS CHKFL0
    310 INTVAR      LDA #$01
    320             STA TLKFL0      ;0=LISTENER / 1=TALKER
    330             RTS
    340
    350 INTVI0      LDA #$93
    360             JSR CHROUT      CLEAR SCREEN
    370             LDA #$00
    380             STA EXTCOL
    390             STA BGCOL0      ;BLACK SCREEN AND BORDER
    400             RTS
    410             ;
    420 BATL0       LDA C12PRA
    430             ORA #$08
    440             STA C12PRA      ;MAKE SERIAL LINE DATA LOW
    450             RTS
    460             ;
    470 BATN1       LDA C12PRA
    480             AND #$BF
    490             STA C12PRA      ;MAKE SERIAL DATA LINE HI
    500             RTS
    510             ;
    520 CLKL0       LDA C12PRA
    530             ORA #$10
    540             STA C12PRA      ;MAKE SERIAL CLOCK LINE LOW
    550             RTS
    560             ;
    570 CLKH1       LDA C12PRA
    580             AND #$EF
    590             STA C12PRA      MAKE SERIAL CLOCK LINE HIGH
    600             RTS
    610             ;
    620 BET81T      LDA C12PRA
    630             CMP C12PRA      AWAIT STABLE CLOCK & DATA LINES
    640             BNE BET81T
    650             ASL A           SET DATA IN  C & CLOCK IN  N
    660             RTS
    670             ;
    680 LINREL      LDA C12PRA
    690             AND #$CF
    700             STA C12PRA      RELEASE DATA & CLOCK (BOTH HI)
    710 LISTEN      SEI
    720 WLOCLK      JSR GET81T      WAIT FOR ATTENTION
    730             BPL WLOCLK      SEND ACKNOWLEDGE
    740             JSR DATL0
    750             JSR DELAY
    760             JSR DATH1
    770             LDX #$08
    780             JSR GET81T
    790 WHICLK      BPL WHICLK      WAIT FOR DATA VALID
    800             ROL BYTE        GET BIT
    810             JSR CLKL0       THEN ACKNOWLEDGE IT
    820             JSR DELAY
    830             JSR CLKH1       RELEASE CLOCK LINE
    840             JSR DELAY
    850             DEX
    860             BNE WHICLK      ;DO EIGHT BITS
    870             CLI
    880             LDA #$0A        LIBAT REM TEXT
    890             STA COLOUR
    900             LDA BYTE
    910             JSR CHROUT      ;PRINT CHAR TO SCREEN
    920             LDA BYTE
    930             CMP #$00
    940             BNE LISTEN      IF RETURN KEY THEN TALK
    950             INC TLKFL0
    960             JMP CHKFLG
    970             ;
    980 DELAY       LDY #$00
    990 WAIT        DEY
   1000             BNE WAIT
   1010             RTS
   1020
   1030 BYTE         BYTE   $00
   1040 TLKFL0       BYTE   $00
   1050 DMDCH1       $00

80  200 DATA 80,85,84,36,60,80,0
    0,73,67,65,32 78,80,77 69,48
    ,1120
43  200 DATA 82,13 17,67 65 03,6
    1,45 32,47,32 60,73 63 75 81
    ,900
AA  300 DATA 32,56,36 78,82,32,5
    7 50,45,32,0,60 85 83 32,82,
    853
9E  310 DATA 79,85,84 69 32,54,6
    2,0 6 0 0 255 255,255,265,0,
    1475
```

DON'T GET LEFT OUT... GET IN ON THE ACTION

COMMODORE DISK USER is a lot more than just another computer magazine. Every issue carries a diskette containing more than £30 worth of software ranging from serious programming utilities to arcade games. There are plenty of Commodore magazines on the market but we believe that this is the *first* to cater for disk users of all ages and tastes.

COMMODORE DISK USER is what you have been waiting for – take out a subscription TODAY!

Please begin my subscription(s) to COMMODORE DISK USER with the issue
I enclose my cheque/money order for £ made payable to Argus Specialist Publications Ltd
or debit £ from my Access/Barclaycard No
valid from to
NAME (Mr/Mrs/Miss)
ADDRESS
Postcode
Signature
Date

Please use BLOCK CAPITALS and include post codes

SUBSCRIPTION RATES

£15.00 for 6 issues U.K.
£18.00 for 6 issues EUROPE
£18.20 for 6 issues MIDDLE EAST
£19.30 for 6 issues FAR EAST
£18.40 for 6 issues REST OF WORLD
Airmail Subscriptioo Rates on Request

Send your remittance to:
INFONET LTD., 5 River Park Estate, Berkhamsted, Herts. HP4 1HL.